Strategic, Tactical and Operational University Timetabling

Lindahl, Michael

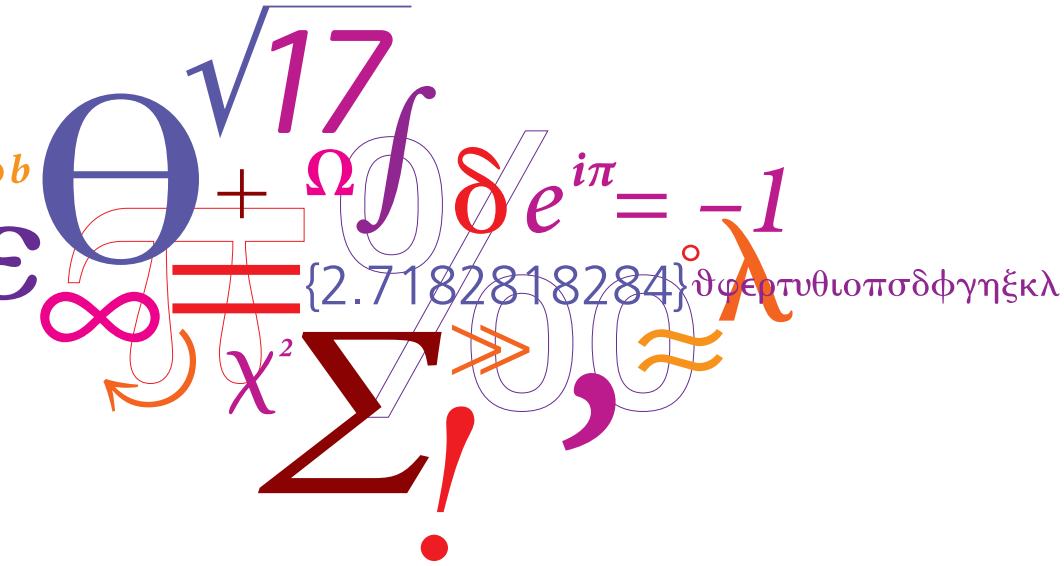
Publication date:
2017

Document Version
Publisher's PDF, also known as Version of record

Strategic, Tactical and Operational University Timetabling

Michael Lindahl

Ph.D. Thesis



Michael Lindahl
Strategic, Tactical and Operational University Timetabling
Ph.D. Thesis, January 2017

Technical University of Denmark
Operations Research Group
Division of Management Science
Department of Management Engineering
Anker Engelunds Vej 1, Bygning 101A
DK-2800 Kgs. Lyngby

MaCom A/S
Vesterbrogade 48,1.
DK-1620 Copenhagen

Abstract

University education is delivered via lectures and classes that are attended by students. When and where these classes are taught is determined by the timetable. A timetable has many stakeholders, and it is the task of planners to accommodate their needs as far as possible, as it has a significant influence on the daily life of both staff and students. Furthermore, it has a large impact on the use of the university's resources. Rooms are a significant cost, and as many are allocated specifically to teaching it is important that the planners optimize their use. Creating a high-quality timetable is, therefore, essential to providing an excellent education, while at the same time using the university's resources efficiently.

This thesis presents an introduction to the university course timetabling problem and its different formulations. Although university timetabling has been widely studied in the literature, work has focused on the creation of a schedule once all of the available resources have been determined, called the *course assignment problem*. This thesis broadens the perspective by also investigating the decision problems that must be solved before and after the course assignment problem. One important problem is to determine the necessary resources. This thesis formulates the *room planning problem* that determines which rooms are available, and the *teaching periods problem* that determines timeslots for teaching. It then analyzes how the available resources affect the quality of the timetable. Once the timetable has been generated, there can be disruptions. This thesis investigates the *quality recovering problem*, which addresses this issue. In this case, an important constraint is that the new solution must be similar to the initial one, but not degrade the quality of the timetable.

Solution methods are presented to these four problems. These are based on mixed-integer programming, and the same underlying model is used in different ways to solve decision problems that occur at different levels of the organization. All methods are tested on the curriculum-based course timetabling problem used for the Second International Timetabling Competition, which is the most-studied problem formulation in the literature.

Finally, this thesis suggests potential directions for future research, which aims to ensure that there are tangible benefits for planners and universities.

Resumé (Danish Abstract)

Undervisning på universiteter foregår ved at de studerende møder op til forelæsninger og gruppeopgaver. Tidspunkt og sted for disse aktiviteter er fastlagt af et skema. Mange mennesker har interesse i skemaet og det er skemalæggernes opgave at imødekomme deres ønsker så meget som muligt, da det har stor indflydelse på den daglige rutine for både undervisere og studerende. Derudover, har det også en stor indflydelse på forbruget af universitetets ressourcer. Undervisningslokaler er en stor omkostning, så det er vigtigt at planlæggerne udnytter dem optimalt. For at kunne have god undervisning er det derfor vigtigt at have skemaer med høj kvalitet samtidig med at universitets ressourcer bruges effektivt.

Denne afhandling giver en introduktion til skemalægningsproblemer på universiteter og dets forskellige formuleringer. Selvom at universitetsskemalægning har været studeret meget, har forskningen mest fokuseret på at lægge hele skemaet når ressourcerne er fastlagt, kaldet *kursusplanlægningsproblemet*. Denne afhandling breder perspektivet ud og undersøger de beslutninger der skal løses før og efter dette problem. Her er et vigtigt problem, at fastlægge hvilke ressourcer der er nødvendige. Der formuleres *rumplanlægningsproblemet* der beslutter hvilke lokaler der bør benyttes, og *undervisningstimeproblemet* der beslutter på hvilke tidspunkter i løbet af ugen undervisning kan foregå. Vi analyserer hvordan tilgængelige ressourcer påvirker kvaliteten af skemaet. Efter skemaet er lagt vil der komme ændringer der gør at noget skal planlægges om. *Kvalitetsrekonstruktionsproblemet* løser det, hvilket også analyseres. Her er det vigtigt at finde en løsning der ligner den forrige uden at gå for meget på kompromis med kvaliteten af det nye skema.

Vi præsenterer løsningsmetoder til alle disse fire problemer. De er alle baseret på heltalsprogrammering og bruger den samme underlæggende model til at løse de forskellige beslutningsproblemer der foregår på alle niveauer i organisationen. Alle metoder bliver testet på det problemformuleringen fra den anden internationale skemalægningskonkurrence, hvilket er det mest studerede problem i litteraturen.

Til sidst gives konklusioner og forslag til retninger for fremtidig forskning, så der kan skabes værdi for skemalæggere og universiteter.

Preface

This thesis was carried out at the Divison of Management Science, Management Engineering at the Technical University of Denmark in partial fulfillment of the requirements for acquiring the Doctor of Philosophy (Ph.D.) degree. The project was supervised by Associate Professor Thomas Stidsen, with Matias Sørensen as co-supervisor.

The project was sponsored by MaCom and the Ministry of Higher Education and Science under the industrial Ph.D. program. MaCom is a software company with 20 employees that provides administration, planning, and communication software to Danish high schools. The study was conducted from February 2014 to January 2017. The project included an extended stay at the University of Auckland, visiting Associate Professor Andrew Mason.

This thesis comprises an introduction and overview of the problem, and three papers that were submitted to international, peer-reviewed journals. It ends with some conclusions and a discussion of the work, together with suggestions for further research.

Kongens Lyngby, Denmark, January 2017

Michael Lindahl

Acknowledgements

A lot of time and effort has been put into this thesis and I am grateful to the people who have helped along the way.

First, I would like to thank my two supervisors, Thomas Stidsen (from DTU Management) and Matias Sørensen (from MaCom) who both have been an essential part of this project. They have given me the freedom to pursue my ideas and have provided valuable input to methods as well as the papers.

I would also like to thank MaCom for giving me the opportunity to pursue a PhD, as well as being great colleagues who were both interested in, and curious about the project. I am also grateful to all my colleagues at the Management Science Department at DTU. Many people have provided input throughout the project and given feedback on the scientific papers before submission.

Andrew Mason and the Department of Engineering at University of Auckland also deserve a big thank you for giving me a warm welcome to New Zealand and ensuring that I had a pleasant stay. It was a great experience working with Andrew and I definitely brought some new perspectives about how to conduct research back home with me. Also, many thanks to the timetable planners at Aarhus University for allowing me to learn about the difficulties they experience.

I would also like to give a special thanks to my colleague and friend Niels-Christian Fink Bagger who put me in contact with MaCom where we started our Ph.D. projects. I appreciate that we have been able to take this journey towards a Ph.D. together, and the many fruitful discussions there have been along the way have definitely helped to make it a great experience. I would like to thank Philippe Jørgensen for helping with the figures - they look great!

Finally, I would like to thank my friends and family for their support. I would also like to especially thank my girlfriend for supporting me, not least during my external research stay on the other side of the world.

Contents

Part I

Introduction

Chapter 1

Background

The importance of education cannot be underestimated. Universities are found worldwide, and they deliver new knowledge and skills. Most teaching is delivered in the form of lectures and classes. During lectures, anywhere from 20 to 200 students are taught as a group. In classes, small groups of approximately 20 students apply their knowledge, supervised by a teacher. When and where these lectures and classes are taught is determined by a timetable, which determines the schedule for all rooms, courses, teaching staff and students.

The timetable has a significant impact on the weekly routine of students and staff as it limits their other activities. Many of a university's resources are allocated to teaching, both in terms of staff time needed to plan and deliver materials, and in terms of space, where universities must allocate rooms for lecture and classes. Therefore, it is important to create high-quality timetables that meet the needs of students and staff, and utilize resources efficiently.

Figure 1.1: The final timetable for each semester is preceded by various stages of information collection and decision-making.

As illustrated in Figure 1.1, the preparation of the final timetable requires multiple steps. Most universities have a semester structure; courses are run either in the spring or in the fall, and a new timetable is drawn up twice per year. The first decision is, therefore, to decide which courses will be offered. Next, planners collect the necessary data, which includes the number of lectures per week, their length, and how many students will attend. The following step is to determine available resources, i.e. which rooms can be used, during

which timeslots. When this is complete, planners can assign courses to timeslots and rooms. Finally, the timetable is published. However, unexpected disruptions can occur and adjustments may need to be made. For example, if a room becomes unavailable during the semester, the planners must find an alternative solution and update the timetable. Universities, like all other organizations, constantly need to make sure that they are using their resources efficiently. Two recent events in Denmark have highlighted the importance of managing the timetable.

First, universities are required to cut annual costs by two percent. One way to achieve these savings is by reducing the number of administrative staff. Consequently, tasks such as timetabling are often centralized and efficiency tools are needed for staff to be able to do the same amount of work in less time. Another consequence of the drive to cut costs is that universities are keen to reduce their usage of rooms and buildings. Optimization tools are therefore needed to ensure the optimal use of space. An example of this comes from Roskilde University, where it was necessary to save 95 million DKK (13M EUR). An analysis from a consultancy firm suggested that savings of 28 million DKK (4M EUR) could be made by reducing the number of rented buildings. Estimating the consequences of terminating a lease for the timetable is, therefore, important to ensure that the best decision is made, and that the timetable remains feasible.

The second change is the Study Progress Reform (*fremdriftsreformen* in Danish). This means that universities have to ensure that all students graduate within a standard timeframe, which has significant consequences for planners. If a student needs to extend their education by an extra semester because two courses were assigned at the same time and it was not possible to follow both, the university receives less money. Ensuring that all students are able to take their required courses in one semester is, therefore, important.

There is an abundant scientific literature on University Course Timetabling. However, most work has focused on a single step: course assignment. How to determine the resources needed and responding to disruptions to the timetable have received far less attention.

Decision support for timetabling planners is limited. Planners often have to juggle competing demands from students, staff and the university administration in trying to draw up a timetable that suits everyone. Therefore, providing them with tools that can help them to prepare better timetables, respond more quickly to changes, and determine the right amount of resources is an important part of ensuring a high educational standard at universities everywhere.

1.1 Thesis Outline

This thesis consists of two parts. Part I introduces the field and the problems that are addressed. Chapter 2 describes the framework and methods used to solve these problems in operations research. Depending on the reader's familiarity with the field and the methods, this chapter (or parts of it) can be skipped. Chapter 3 gives an introduction to the university course timetabling problem and how it relates to other problems. In Chapter 4 the problem is analyzed from strategic, tactical and operational perspectives. Chapter 5 summarizes these contributions and offers some overall conclusions drawn from the work presented in this thesis, together with perspectives for future research.

Part II consists of three scientific papers. These papers address the research topics

and constitute the majority of the work in this thesis.

Chapter 2

Operations Research

Operations research is the discipline of applying advanced analytical methods to help make better decisions (Informs [2017]). It originated in military planning during the two world wars and was later extended to a wide range of industries for purposes such as logistics, network design, finance, and of course timetabling.

The framework used by the operations analyst is illustrated in Figure 2.1 and the following presents its use in the context of university course timetabling.

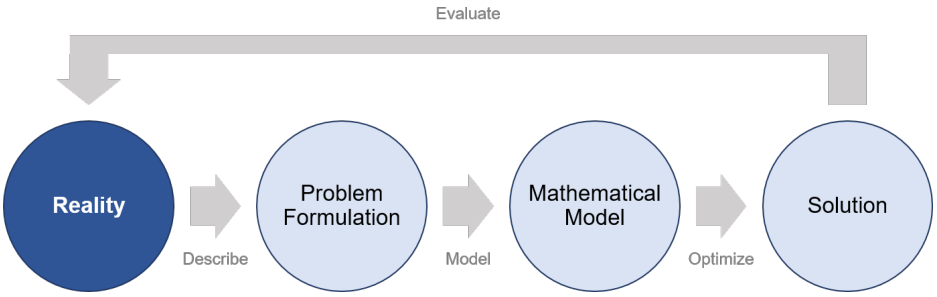


Figure 2.1: Operations Research develops a mathematical representation of reality to improve decisions.

A problem always starts in reality. For example, an organization that must solve the problem of creating good, feasible timetables. The approach begins with an analysis that aims to formulate the problem, by stating what the objective is, and what the decisions that need to be taken are. At this stage, it is important to correctly identify the scope: for example, it may be necessary to decide which room should be used for a course and at what time, but not which courses should be offered in a particular semester. It is also important to identify any constraints, for example, that courses followed by the same students cannot be taught at the same time.

Once the problem has been formulated, a mathematical model is developed that describes the problem in variables and equations. A mathematical optimization model consists of three elements:

Variables What are the decisions that need to be made? For example, when should a course be planned?

Objective Given a set of decisions, how good is the solution? For example, it may be necessary to limit courses taught in the evening.

Constraints What rules needs to be followed for a solution to be feasible? For example, students can only attend one course at a time.

This leads to an unambiguous formulation of potential solutions that can be compared and evaluated to identify which is best.

Once the mathematical model has been developed, the next goal is to identify the optimal value of decision variables that meet all of the model's constraints. This is known as optimization, and there are a wide selection of methods that are discussed in the next section.

When a solution is found, it is compared with reality and evaluated. In this case, a solution is a timetable that describes when all courses should be taught. This timetable can be presented to planners who can evaluate it. If it is acceptable, it can be published, on the other hand, it may not be usable. For example, a lecture may need to be given in a room with specific equipment. In this case, the operations analyst needs to adjust the problem formulation and mathematical model to include the new requirements.

2.1 Optimization Methods

Optimization is about finding the best solution to a mathematical model that must either be maximized or minimized. This section briefly describes some methods to find this solution.

Finding the optimal solution can be challenging, as the number of potential solutions is often too large to search through them all. For example, a timetabling problem can involve 100 lectures that can be assigned to 20 timeslots, and five rooms. This problem will have $100! \approx 10^{157}$ potential solutions. Optimization methods can be divided into two types: heuristic and exact.

Heuristics are algorithms that search the solution space using a strategy that seeks to find a good solution without any guarantee that it is optimal. Generic heuristic frameworks are called metaheuristics. An example of a metaheuristic is the *hill climber*, which starts with a solution and iteratively improves it by finding the best one within a defined neighborhood (e.g. by swapping the timeslots for two courses). Other examples of two widely-used metaheuristics are Tabu Search (Glover and Laguna [2013]) and Simulated Annealing (Laarhoven and Aarts [1987]).

Exact methods guarantee that a mathematically-optimal solution to a given problem is found. One method that has been used with success on a large variety of problems is *mixed integer programming*.

2.1.1 Mixed integer programming

Mixed integer programming (MIP) is a flexible and popular method. It is an extension of linear programming, in which a mathematical model consisting of linear constraints

and an objective is formulated. The difference between linear programming and MIP is that in the former all variables have to take continuous values, while in the latter some can take integer values. Let $\mathbf{x}$ be a vector of decision variables. Then, let $\mathbf{b}$ and $\mathbf{c}$ be a vector of real numbers, and A be a matrix of real numbers. A MIP model then takes the following form:

$$\begin{aligned} \min \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} && (2.1)\\ \text{s.t.} \quad & A\mathbf{x} = \mathbf{b} && (2.2)\\ & \mathbf{x} \geq 0 && (2.3)\\ & \text{some } \mathbf{x} \text{ only take integer values} && (2.4) \end{aligned}$$

Problems in this form can be solved using a branch-and-bound method, described in Land and Doig [1960]. For minimization problems, the method operates with two values for the objective, a lower bound, and an incumbent. The incumbent is the best feasible solution found so far, and this value will decrease over time as a better solution is found. The lower bound is then the minimum potential value of the objective. This value increases during the search; when the lower bound and the incumbent reach the same value, the optimal solution is found. If the method is aborted before the optimal solution is found, the solution is still bounded, meaning that no matter how much longer the algorithm is run, no better solution will be found.

Software packages exist to solve such MIP models. These solvers include many advanced features such as pre-solve, cutting planes and matheuristics, in addition to the branch-and-bound method. Consequently, as demonstrated by Bixby [2012], there has been an overall improvement in the performance of the performance of a factor of over 29,000 since the early 1990s. Finally, computing power has increased exponentially, making this a powerful tool for operations analysts.

2.1.2 Matheuristics

Matheuristics refers to the combination of mathematical programming methods such as MIP, and heuristic methods.

As described in Lodi [2013], MIP solvers take many heuristic decisions to solve models. One example is the branch-and-bound procedure, where the solver needs to choose which variable to branch on. Solvers also include heuristics to find and improve solutions more quickly; so-called *primal* and *improvement* heuristics. Primal heuristics help to find feasible solutions faster. One example is the feasibility pump proposed in Fischetti et al. [2005]. Here, the solver uses fractional node relaxation to search for a nearby integer-feasible solution. Improvement heuristics improve an already-found solution. One example is the Relaxation Induced Neighbourhood Search (RINS) proposed in Danna et al. [2005]. The solver takes a fractional node relaxation, fixes variables that have the same value as the incumbent, and solves that. For an overview of primal and improvement heuristics, see Berthold [2006].

2.2 Multi objective optimization

In multi objective optimization, there are several objectives rather than one. This is often the case in timetabling, where multiple attributes must be minimized. A multi objective optimization problem with two objectives, f_1 and f_2, is called a bi-objective optimization problem. It is stated using two objective vectors $\mathbf{c}_1$ and $\mathbf{c}_2$, which are used to calculate the two objectives:

$$\min \quad f_1 = \mathbf{c}_1^{\mathrm{T}}\mathbf{x} \tag{2.5}$$

$$f_2 = \mathbf{c}_2^{\mathrm{T}}\mathbf{x} \tag{2.6}$$

New methods are required to solve this. One way to overcome this problem is to combine the two objectives into one, by introducing the parameters $w_1, w_2 \in \mathbb{R}$ and calculating the weighted sum of the two objectives:

$$\min \quad w_1 f_1 + w_2 f_2 \tag{2.7}$$

The problem with this method is that a linear function might not be able to represent how the planner makes a trade-off between the two objectives. Rather than giving the planner one solution, it is better to generate multiple solutions and let them choose which one they find best.

Pareto-optimal solutions Figure 2.2 illustrates how to identify multiple solutions by generating a front of so-called *pareto-optimal solutions*. A pareto-optimal solution is one where one objective cannot be improved without degrading the other. A solution is termed *dominated* if there exists another solution where both objectives are better. Solutions that are not dominated are part of the pareto front indicated by the line. In the example, the solution x_1 is better than x_2 for the objective f_1, but x_2 is better for f_2. The two corner solutions where, respectively, f_1 and f_2 are minimum, are called the lexicographic solutions. These are used if objectives can be prioritized, for example if f_1 is the priority and the aim is to minimize it, and at the same time f_2 can be minimized without degrading f_1. Different methods exist to generate pareto-optimal solutions.

Epsilon Method One way to find pareto-optimal solutions is the epsilon method, proposed in Haimes et al. [1971]. This introduces the parameter $\epsilon \in \mathbb{R}$. The method then consists of selecting an objective, turning it into a constraint, and using ϵ as an upper bound to solve the new problem, which results in a pareto-optimal solution.

$$\min \quad f_1 \tag{2.8}$$

$$\text{s.t.} \quad f_2 \leq \epsilon \tag{2.9}$$

The lexicographic corner points can be found by solving two optimization problems. The first is to find the minimum value of the highest-priority objective by removing the other objective and optimizing. The next is to turn the primary objective into an epsilon constraint, with an upper bound corresponding to the value of the objective, and optimizing for the second objective.

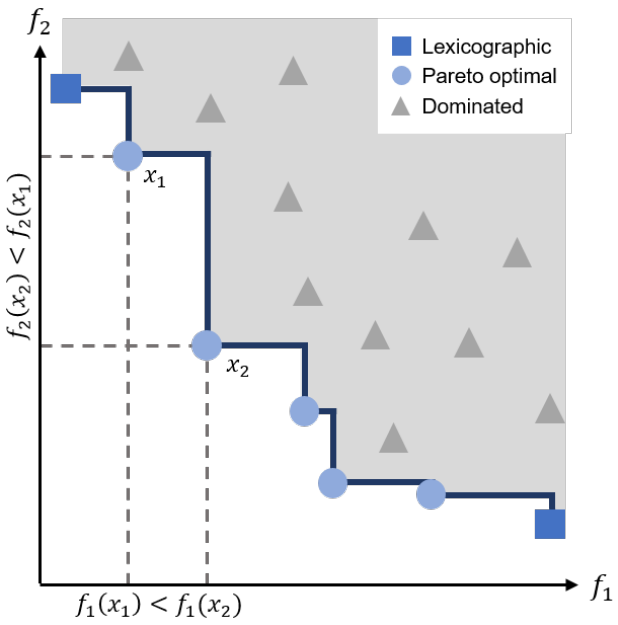


Figure 2.2: The pareto front consists of solutions where no other solutions are better given both objectives. The two lexicographic solutions are found in the corners of the pareto front.

Chapter 3

University Course Timetabling

In university course timetabling problem, the goal is to assign lectures to rooms and timeslots given various hard and soft constraints. In this chapter, we define university course scheduling and present the different formulations.

Figure 3.1 illustrates the families of scheduling problems, which Wren [1996] defines as:
"the allocation, subject to constraints, of resources to objects being placed in space-time, in such a way as to minimize the total cost of some set of the resources used."

Scheduling problems include transport scheduling (e.g. Laporte [1992]) and job shop scheduling (e.g. Taillard [1993]).

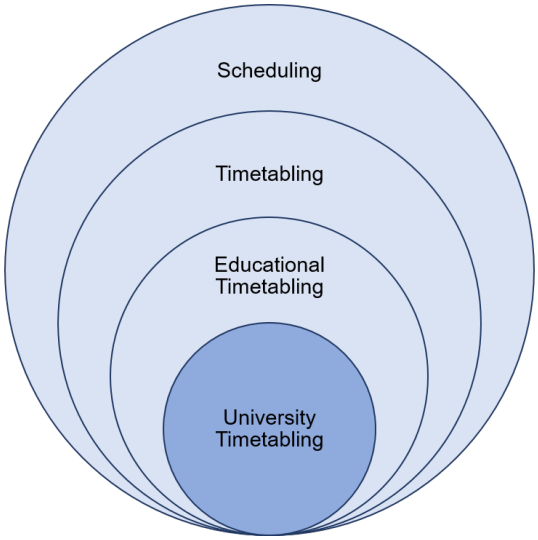


Figure 3.1: University timetabling is a timetabling problem that is part of the broader family of scheduling problems.

The timetabling problem is part of this broader family of scheduling problems, and Wren [1996] defines it as:

"the allocation, subject to constraints of given resources to objects being placed in space-time, in such a way as to satisfy as nearly as possible a set of desirable objectives."

The category includes Employee Timetabling (e.g. Meisels and Schaerf [2003]), Rostering problems (e.g. Burke et al. [2004]) and Sports Timetabling (e.g. Kendall et al. [2010]). A large subset of these problems concern what is called Educational Timetabling (e.g. Kingston [2013], Kristiansen and Stidsen [2013]), which includes timetabling problems related to education at all levels, for example high schools (e.g. Kingston [2010], Sørensen and Stidsen [2012]). University timetabling refers specifically to solving problems related to universities and there have been several comprehensive overviews (Bardadym [1996], Burke et al. [1997], Carter [2001], McCollum [2007], Lewis [2008], Phillips [2015]). The usual definition refers to three problems, illustrated in Figure 3.2.

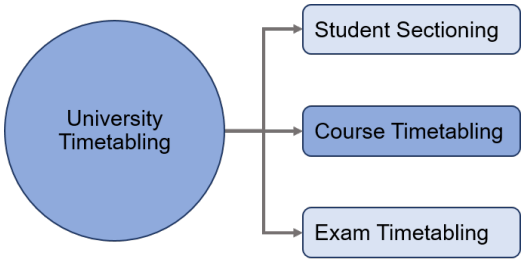


Figure 3.2: The three university timetabling problems.

The first is the student sectioning problem (e.g. Müller and Murray [2010]), where courses are offered at various times during the semester to allow more students to follow them; here the goal is to assign students to different sections. The second is the course timetabling problem, which has received the most attention. This problem has several variations that are described in Chapter 4. The third is the exam timetabling problem (e.g. Carter [1986]), where the goal is to schedule exams for all courses.

The Course Timetabling Problem In the course timetabling problem, illustrated in Figure 3.3, courses must be assigned to timeslots and rooms (input), leading to an output, termed *assignments*. The form of these assignments is a function of hard and soft constraints. Hard constraints must be satisfied in order for a timetable to be feasible, for example, two courses cannot be held in the same room. Soft constraints distinguish between a good and a bad timetable. An example could be that teachers prefer not to teach in the morning.

Variants of this problem only aim to assign timeslots or rooms. For example, at the Technical University of Denmark and the University of Auckland (New Zeland), timeslots are decided by departments, and rooms are assigned later, at the institutional level. The timeslot assignment is studied in Lach and Lübbecke [2008], while the room assignment problem is studied in Phillips et al. [2015].

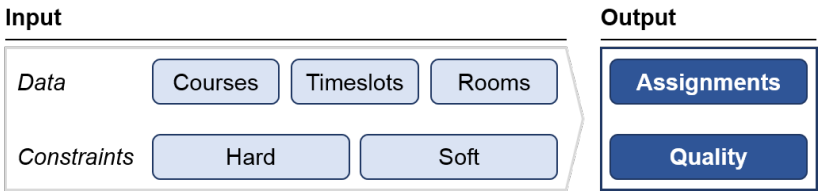


Figure 3.3: The course timetabling problem aims to assign courses to timeslots and rooms while taking hard and soft constraints into account, in order to generate the optimal timetable.

3.1 Hard and soft constraints

There are significant differences between universities in the hard and soft constraints that are taken into account when creating timetables. These differences are due to educational differences, organizational culture, and regulations. A soft constraint at one university can be a hard constraint at another (MirHassani and Habibi [2013]). For example, the University of Udine (Italy) allows rooms to be filled beyond capacity, because some students drop out at the beginning of the semester. But at the University of Auckland, fire regulations prevent a room being filled beyond its capacity.

3.1.1 Hard Constraints

Most hard constraints are similar between universities and are covered in Di Gaspero et al. [2007], Lewis et al. [2007], Phillips et al. [2015].

Lectures All lectures for a course should be assigned a timeslot and a room.

RoomOccupancy Only one lecture can take place in one room in one timeslot

Conflicts Two lectures taken by the same student or taught by the same member of staff cannot be assigned to the same timeslot.

Availability If a teacher cannot teach in a specific timeslot, no lectures taught by that teacher can be assigned to that timeslot.

RoomFeatures The room must have any necessary equipment (e.g. a lab for chemistry).

Temporal Courses have constraints regarding how they relate to each other. For example, they may share an external lecturer and therefore should be run in consecutive timeslots.

Although this list is not exhaustive, it indicates the range of hard constraints.

3.1.2 Soft Constraints

Soft constraints determine what a good timetable looks like, and differ between universities. They are always tied to one of three stakeholders: staff, students or management,

who have different and sometimes even conflicting goals. Staff care about having an appropriate room for their lectures, students care more about having a timetable that suits their lifestyle, while management cares about the optimal use of rooms (Bonutti et al. [2012], Phillips et al. [2015] and MirHassani and Habibi [2013]). Here, we investigate soft constraints with respect to these three groups of stakeholders.

Staff

Teaching staff care about having the right environment to teach their course. Another significant factor is their working environment; for example they may prefer to carry out research rather than teach on specific days.

RoomCapacity The room should be big enough to accomodate all students.

RoomSuitability The room should be suitable for the teaching methods, e.g. some lectures include group work and the room should, therefore, be able to accommodate this.

RoomStability If there are multiple lectures in the same week, they should be assigned to the same room.

TimeSuitability The lecturer may have a preferred timeslot for teaching.

Temporal If there are both lectures and group exercises, the lecture should be held before the exercises.

Fairness Attempts should be made to respect the preferences of all members of staff equally.

Students

The primary concern of students is to have a timetable that supports their learning and does not conflict with their other activities.

CourseWishes Students create a wishlist of the courses they wish to follow and it is the planner's task to ensure that most requests are met.

TravelDistance When students have multiple lectures in one day, they should be assigned to rooms that are physically close to each other.

MinWorkingDays A course that has many lectures during the week should be spread over a minimum number of days.

CurriculumCompactness When students have multiple lectures in one day they should be run one after the other to avoid unproductive empty timeslots.

Lunchbreaks Students should have a lunch break around noon.

Workload The daily workload should be averaged out over the week.

Management

The priority for management is to utilize resources efficiently. Chapter 4 examines management decisions in detail. One measure used in course assignment is:

RoomUtilization Empty rooms should be placed in consecutive timeslots so that they can be used for other purposes (e.g. conferences).

3.2 Formulations

A *formulation* of university timetabling refers to a version of the course assignment problem that includes a selection of soft and hard constraints. As formulations usually contain multiple soft constraints, they also include a weighting system that associates each soft constraint with a number of penalty points in order to be able to compare two solutions and identify the best. Formulations differ between universities. Many examples can be found in the literature: Hochschule Konstanz by Erben and Keppler [1996], the University of Valencia by Alvarez-Valdes et al. [2000], Athens University of Economics and Business by Dimopoulou and Miliotis [2001], Ohio University by Martin [2004], the University of Udine by Di Gaspero and Schaerf [2006], Purdue University by Murray et al. [2007], TU-Berlin Lach and Lübbecke [2008], and the University of Auckland by Phillips et al. [2015].

As researchers were working on different problems, it became clear that it was not possible to compare results and methods. To overcome this, an international timetabling competition was held in 2002 (Network [2002]). The organizers proposed a problem formulation, together with a selection of instances, and participants competed to find the best solutions. Five years later, in 2007, the Second International Timetabling Competition (ITC2007) was held that included two formulations of the university timetabling problem: the curriculum-based timetabling problem, formulated in Di Gaspero et al. [2007], and the post-enrollment timetabling problem, formulated in Lewis et al. [2007]. The biggest difference between the two formulations is that individual students are not considered in the curriculum-based version. Instead, they are taken into account through curricula, where a curriculum is a set of courses followed by some students. The post-enrollment version considers each student individually, and attempts to ensure that each has a high-quality, feasible timetable. Later, Bonutti et al. [2012] proposed five variations of the curriculum-based problem, including the original formulation. The goal was to develop a more diverse set of benchmark datasets, and encourage researchers to reduce their particular problem to one of these five, thereby making it possible to compare results and methods.

3.2.1 Curriculum-based Timetabling

The curriculum-based course timetabling problem (CB-CTT) is the most widely-used formulation for university course timetabling. It was initially proposed in Di Gaspero et al. [2007] in the context of ITC2007.

In CB-CTT the goal is to assign all courses to timeslots and rooms in a one week period. The week consists of days (usually 5 or 6) and each day consists of timeslots (usually also 5 or 6). Each course consists of a number of lectures, which corresponds to

the number of timeslots for the week. Each course is associated with a member of staff, and a number of students who attend it. A set of rooms is identified, and each room has a capacity that corresponds to the number of people it can accommodate. Finally, the curricula is the set of courses attended by students. The problem includes the following hard constraints:

Lectures All lectures should be scheduled.

RoomOccupancy A room can only accommodate one lecture at a time.

Conflicts Lectures that are part of the same curriculum, or taught by the same member of staff cannot be assigned to the same timeslot.

Availabilities Each member of staff has a set of timeslots during which he or she is unavailable to teach.

Soft constraints are as follows, and each is associated with a number of penalty points:

RoomCapacity The room should be able to accommodate all students. Each student above this capacity attracts one penalty point.

MinimumWorkingDays Each course has a minimum number of working days it should be spread over. There is a five-point penalty for each day below this.

CurriculumCompactness Lectures that are part of the same curriculum should be assigned to consecutive timeslots. For each curriculum, two penalty points are awarded to each lecture that does not have a lecture from the same curriculum either before or after it.

RoomStability All lectures that are part of the same course should be taught in the same room. Each extra room used for a course attracts one penalty point.

The objective is then to find a feasible solution that respects all of the hard constraints and minimizes the number of penalty points.

Twenty-one instances were provided, shown in Table 3.1. These varied in both the number of courses and the number of curricula. More instances were defined in Bonutti et al. [2012], and subsequently Mühlenthaler defined six, very large instances from the University of Erlangen. An initial instance generator was proposed in Burke et al. [2010], followed by a more advanced version in Lopes and Smith-Miles [2010, 2013]. The latter is able to generate more realistic instances by utilizing more advanced statistical methods. In Burke et al. [2010] they show that the problem is $\mathcal{NP}$-hard as the hard constraints create a *graph coloring problem*.

To help researchers to compare their results, a website was created by Alex Bonutti. The site contains all of the instances, which can be downloaded. It also allows researchers to upload their solution and validate it for violations of hard and soft constraints. The team track the best-known solutions and lower bounds. The site can be found at `http://tabu.diegm.uniud.it/ctt/` and is maintained by Luca Di Gaspero and Andrea Schaerf. The competition and the website have had a significant influence on research and there is an ongoing search to find better solutions and better lower bounds.

Instance	Courses	Rooms	Timeslots	Curricula
comp01	30	6	30	14
comp02	82	16	25	70
comp03	72	16	25	68
comp04	79	18	25	57
comp05	54	9	36	139
comp06	108	18	25	70
comp07	131	20	25	77
comp08	86	18	25	61
comp09	76	18	25	75
comp10	115	18	25	67
comp11	30	5	45	13
comp12	88	11	36	150
comp13	82	19	25	66
comp14	85	17	25	60
comp15	72	16	25	68
comp16	108	20	25	71
comp17	99	17	25	70
comp18	47	9	36	52
comp19	74	16	25	66
comp20	121	19	25	78
comp21	94	18	25	78

Table 3.1: The 21 instances used for ITC2007, defined in Di Gaspero et al. [2007]. These intstances differ in both size and number of curricula.

Three examples demonstrate how the original goal of the competition has been fulfilled. The first is Lach and Lübbecke [2012], who use the problem to show the applicability of their earlier work (Lach and Lübbecke [2008]). Second is Phillips et al. [2015], who present their work on a specific problem from the University of Auckland, but also use the problem to benchmark their method. The third example is in Mühlenthaler and Wanka [2016], who show how fairness can be included in timetabling problems, and use CB-CTT as the underlying problem formulation.

As McCollum et al. [2010] argue, the competition has been highly successful in bringing the community together; it has introduced new ideas and created interest in the field.

3.3 Previous methods

Both heuristics and exact methods have been applied to the CB-CTT problem. Hertz [1991] uses a Tabu Search. A memetic algorithm is used in Paechter et al. [1995]. An early overview of methods applied to timetabling can be found in Burke et al. [1997]. Timetabling was solved with MIP in Ferland and Roy [1985] and Burke et al. [2007]. Burke et al. [2008b] highlight that taking soft constraints into account makes the problem difficult. This is also shown in Glassey and Mizrach [1986] where a MIP model is proposed,

but the model is solved using a heuristic (due to the difficulty). In Phillips et al. [2015], MIP is used to solve a large-scale practical classroom assignment problem.

One of the rules of ITC2007 was that solvers were not allowed, which resulted in a focus on heuristics. The winning algorithm is described in Müller [2009], and uses both Iterative Forward Search and a Hill Climbing algorithm. Later, two new heuristics were shown to perform better. A Tabu-based memetic approach was proposed in Abdullah and Turabieh [2012], and an Adaptive Large Neighborhood Search in Kiefer et al. [2014].

MIP has also been applied to CB-CTT. First, Burke et al. [2008a] proposed the monolithic model, also known as the three-index model. Later, Lach and Lübbecke [2012] proposed a two-index model, where the the problem is decomposed into two stages that are solved sequentially. The first stage creates the time assignment, and the second creates the room assignment. Finally, MIP has been successfully used to create lower bounds. Hao and Benlic [2011] use a partition-based approach based on the "divide and conquer principle". Cacchiani et al. [2013] split the objective function into two parts and solved each separately using a column generation procedure. For a comprehensive overview and comparison of the methods used to solve the curriculum-based course timetabling problem problem see Bettinelli et al. [2015].

Chapter 4

Strategic, Tactical and Operational Course Timetabling

The course timetable process consists of many decisions. Before courses can be assigned to rooms and timeslots, decisions need to be made regarding what rooms are available and what timeslots can be used. As described previously in Section 3.1, there is a wide selection of constraints, and decisions need to be made regarding which should be used and their priority. Next, the courses must actually be assigned, and after that any disruptions must be managed. These decisions are taken at different levels in the organization, and this chapter describes the different timetabling problems at each level. Finally, we analyze how these problems relate to the values of the organization, and the people involved.

Decision problems can be divided into three organizational levels: strategic, tactical and operational, illustrated in Figure 4.1. Strategic problems define the overall priorities when creating a timetable, and what resources are available. Tactical problems concern how the timetable is drawn up (e.g. processes and priorities). Finally, operational problems concern the actual timetable (i.e. deciding which courses should be in what room and when). We investigate the problems at each level.

4.1 Operational

The main problem here is the *assignment problem* which is described in Chapter 3, given a number of hard and soft constraints. As highlighted earlier, this problem has received most attention in the literature, and is investigated in Chapter 6.

A reduced version of this problem, where only hard constraints are considered, is called the *feasibility problem* (Rudová et al. [2011]). The purpose is to detect if the problem is infeasible due to inconsistencies in hard constraints. Examples include errors in the data or conflicting staff requirements. These checks can be made during the data-gathering phase and help to quickly correct or remove the problem.

After a timetable has been drawn up and published, changes will necessarily occur. Staff may be unable to teach in a certain timeslot, or a new course may need to be assigned. The objective here is to find a new timetable that does not require too many changes to the existing timetable. This leads to the *minimum perturbation problem*, where the goal is to make an infeasible schedule feasible by making as few perturbations (changes) as possible. This problem is investigated using heuristics in Barták et al. [2003], Muller et al. [2005], Rudová et al. [2011]. Another paper by Phillips et al. [2016] uses MIP to create a neighborhood with a part of the solution space, and gradually expand it until a feasible solution can be found.

However, a minimum perturbation solution will often lead to violations of soft constraints, as shown in Chapter 7. It therefore becomes necessary to introduce additional perturbations to reduce the number of violations. This results in the *quality recovering problem*, where soft constraints are taken into account in order to find a solution with the right trade-off between the number of perturbations and the number of violations of soft constraints. This problem is investigated in Chapter 7.

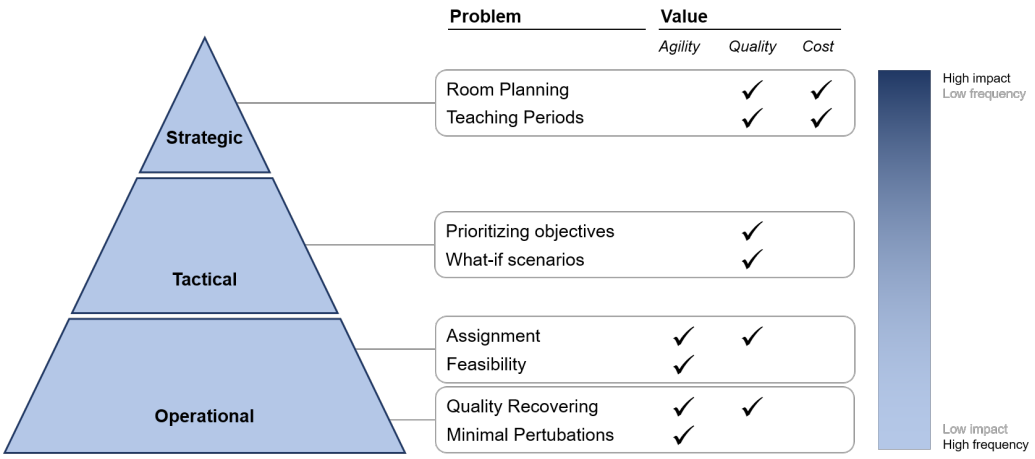


Figure 4.1: Problems are found at different levels in the organization, which impact the final timetable. (figure from Chapter 8)

4.2 Tactical

Tactical problems concern the processes and procedures that planners follow. Most universities have a team of planners who divide the work between them. This will often be at a department or faculty level and, usually, requires them to divide the available rooms between them. Large rooms are often in high demand, and decisions need to be made regarding priorities. This collaboration problem is discussed in Dimopoulou and Miliotis [2004], who note that it is important to take the distributed environment into account when developing a successful planning tool. Such tools can create a flexible, automated

procedure where courses that are shared between departments are planned first, before each department plans their full timetable. Beyrouthy et al. [2010] analyze the consequence of mixing different space types (e.g. lecture rooms and tutorial rooms) using MIP. This study shows how relaxing a hard constraint can increase room utilization.

Another decision is how to weigh different soft constraints. This is known as the *prioritizing objectives problem*. Phillips [2015] discusses how a mathematical model of the assignment problem can be used to generate scenarios for mid-term planning (e.g. analyzing the potential of relaxing hard and soft constraints to evaluate the impact on the quality of the timetable). Silva et al. [2004] provide an overview of heuristics for multi-objective optimization in timetabling and scheduling that can be used to find trade-offs between different objectives.

4.3 Strategic

Strategic decisions encompass all of the high-level, long-term decisions that impact timetables and the organization. In the university context, it concerns available resources. Compared to operational problems, these problems have not received much attention in the literature.

Rooms and time are the two priorities for management. Issues concerns the number and type of rooms and buildings to be constructed, and when students and staff are available for teaching. The *room planning problem* was first investigated in Fizzano and Swanson [2000], who applied a heuristic to the assignment problem and used this to investigate how many rooms could be removed while still retaining a feasible schedule. Later, Beyrouthy et al. [2007] used a greedy algorithm to build a room profile. Beyrouthy et al. [2009] also took a strategic view in their analysis of how respecting soft constraints can drive down room use based on bi-objective optimization. This *teaching periods problem* had not previously been studied in the literature. Both of these problems are investigated in Chapter 8.

4.4 The value of better decisions

Solving these problems, and making better decisions, reflects different values depending on the problem that is solved. We divide these values into three categories: agility, quality and cost, illustrated in Figure 4.1. The following paragraphs discuss these values and show how the different problems contribute to them. It should be noted that this categorization does not mean that the problems do not reflect other values, rather it illustrates the principal values.

Agility Agility is being able to respond to changes, and quickly make alterations to the timetable. All operational problems contribute to this. Being able to make operational decisions faster reduces the stress for planners, by allowing them to find feasible solutions quickly.

Quality A high-quality timetable provides a good working environment for staff and students. This attribute is achieved by taking into account soft constraints that reflect the timetable preferences of students and staff. Both the *quality problem* and the *quality recovering problem* take this into account, as unwanted features are minimized. Tactical problems make a similar contribution. Prioritizing objectives ensures that the right trade-offs are made in the planning process, resulting in high-quality timetables.

Cost A timetable that uses expensive resources efficiently can help the profitability of a university. The capacity of a university is determined by the number of rooms, and the teaching periods. Buildings and rooms are a high cost. Building new capacity is expensive, while unused rooms can often be turned into offices or rented out for profit. Minimizing overcapacity, as described in the *room planning problem* can thereby reduce costs. Another way to increase capacity is to extend the number of teaching periods, for example, by holding lectures later in the evening, as described in the *teaching periods problem*.

As Figure 4.1 illustrates, problems at the top levels have a high impact, but are not often solved. On the other hand, problems at the bottom level are often solved, but their effect is short term and therefore have less impact.

4.5 Problems

This thesis investigates four problems: two are strategic problems, the *teaching periods problem* and the *room planning problem*, and two are operational problems, the *assignment problem* and the *quality recovering problem*.

These problems and the relations between them are illustrated in Figure 4.2. The two strategic problems are solved in order to decide how many of the two resources (timeslots and rooms) are needed. These problems take courses, and hard and soft constraints as input. The two outputs (decisions) concern timeslots and rooms. The third output relates to quality, as more resources increase quality. The decision-maker, therefore, needs to make a trade-off between resources and quality.

Once resources have been determined, the *assignment problem* can be solved. The solution to the assignment problem concerns which timeslot and room each course should be taught in. The objective is to minimize the violation of soft constraints to obtain a high-quality timetable. Once the timetable has been published, disruptions must be managed. This results in the *quality recovering problem*. This problem takes the previous assignments, and the disruption that makes these assignments infeasible as input. The aim is to find a timetable similar to the existing one. A delta function measures how many perturbations there are between the initial solution and the new one. The output is a new timetable with assignments, a quality level, and the number of perturbations from the initial solution.

It should be noted that the initial, strategic problems have fewer inputs and constraints than subsequent problems. The problem is increasingly more constrained by previous decisions; the quality recovering problem is the most constrained and only affects a few assignments.

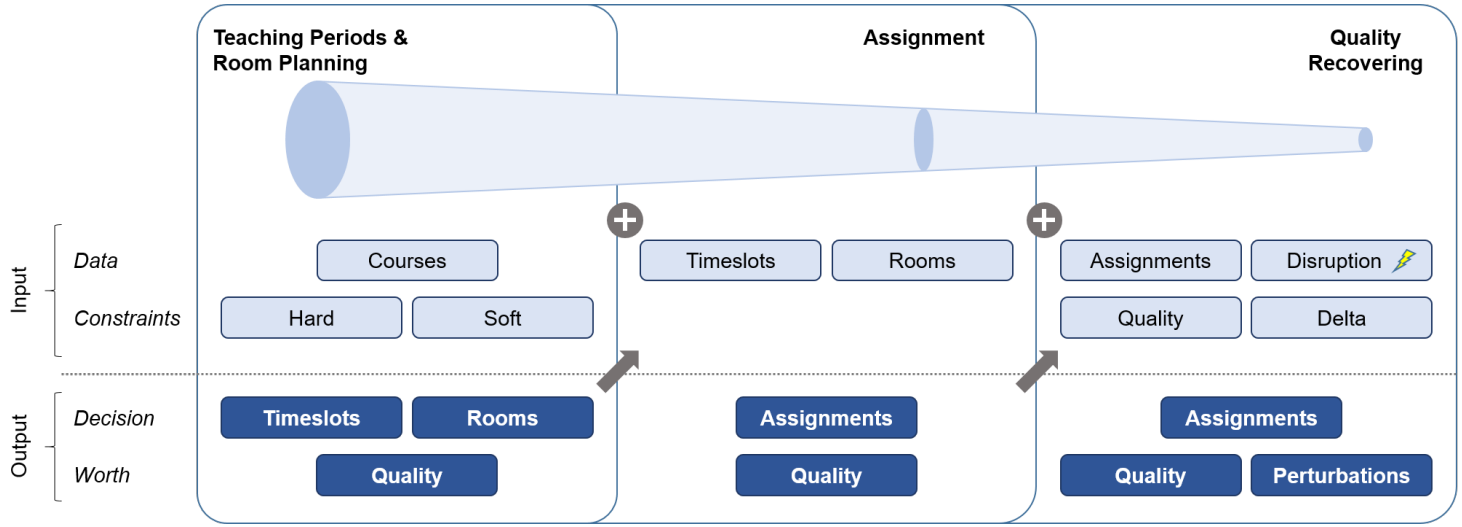


Figure 4.2: Inputs and outputs of the problems. The output from each step is used as input to the next. The first, strategic, problem is to identify the two resources *timeslots* and *rooms*. Next is the *assignment problem*, where the aim is to create a high-quality timetable. Finally, when disruptions occur the timetable must be changed to find a new solution that is similar to the old one.

Chapter 5

Contributions and Conclusions

The scientific contributions of this thesis are contained in three papers that have been submitted to international, peer-reviewed journals. Two of these papers analyze operational *assignment* and *quality recovering* problems, while the third takes a strategic perspective and analyzes the *room planning problem* and the *teaching periods* problem. This work uses the curriculum-based course timetabling (CB-CTT) formulation to define hard and soft constraints, while datasets are taken from the second international timetabling competition (ITC2007) competition. All solution methods are based on mixed integer programming (MIP). This chapter begins with an overview of the contributions of each paper, and ends with overall conclusions and suggestions for further research.

Chapter 6: A Fix-and-Optimize Matheuristic for Curriculum-Based Course Timetabling This paper proposes a matheuristic to find feasible solutions to the course assignment problem. As discussed previously in this thesis, most of the previous work regarding CB-CTT is based on metaheuristics. One reason for this is that the assignment problem is computationally hard to solve using a standard MIP solver. The proposed algorithm combines MIP and heuristics by using a MIP solver to explore different parts of the solution space. The smaller solution spaces are created by fixing a number of variables, while an adaptive layer is used to choose the best neighborhood size to explore. This method generates solutions faster, especially on large instances and performs better than the original winning metaheuristic from the ITC2007 competition. The paper was submitted to the *Journal of Heuristics* and its contributions are:

- The first matheuristic to solve the university course timetabling problem.
- A method to solve very large university course timetabling instances using MIP.

Chapter 7: Quality Recovering of University Timetabling This paper investigates how to recover from disruptions to a feasible timetable. Previous work has solved this problem by searching for a minimum perturbation solution that is as similar as possible to the existing solution. However, this paper shows that such an approach tends to result in a decrease in quality of between 7% and 232% on the ITC2007 instances. It therefore

is beneficial to use additional perturbations in order to find a solution that remained close to the initial one, but with better quality. By allowing more perturbations, the quality decrease in the objective can be reduced to between 2% and 88%. The proposed method uses bi-objective optimization to give the planner a choice of solutions. All solutions are optimal, and as the method is based on MIP it can easily be incorporated into a system (with the fix-and-optimize matheuristic) as it does not make any assumptions about the formulation. The paper was submitted to *OR Spectrum* and its contributions are:

- The first analysis of how minimum perturbation solutions affect the quality of the timetable.
- The first method to find solutions that both take quality and perturbations into account. The proposed method finds optimal solutions using bi-objective optimization.

Chapter 8: A Strategic View on University Timetabling The paper analyzes how available resources affect the quality of the timetable using bi-objective optimization. Decisions concerning available resources have a long-term impact on timetables and are therefore important. The paper also demonstrates the trade-off between room profiles and the number of timeslots needed to create a feasible timetable. As the method is based on MIP, any formulation can be used as the underlying timetabling problem. The problem is computationally hard and the paper also explores why. The paper was submitted to the *European Journal of Operations Research* and its contributions are:

- The first MIP model of the strategic room planning problem.
- The first formulation of the strategic teaching periods problem including a MIP model.
- The first exact method to find the pareto fronts in the trade-off between room profile, number of timeslots and the quality of the timetable.

Published source code To encourage further research in this area the source code has been made publicly available at `http://github.com/miclindahl/UniTimetabling`. Specifically, this includes the codebase used in the computations given in the two papers *A Strategic View of University Timetabling* and *Quality Recovering of University Timetables*. A flexible framework is provided that makes it easy to test different formulations, algorithms and parameters. The codebase includes among others:

- A datareader and solution validator to CB-CTT.
- Two implemented MIP models: The 3-index from Burke et al. [2008a] and the 2-index from Lach and Lübbecke [2012].
- Three additional objective measures: total seats, number of timeslots and perturbations.
- An implementation of the epsilon-method for bi-objective optimization that finds trade offs between the objectives.
- A comprehensive test framework to compare the impact of different parameters.

5.1 Conclusion

University Course Timetabling has received a lot of attention in the literature. However, most of this work has focused on the *assignment problem*. This thesis takes a broad perspective, by exploring the decisions that are made at strategic, tactical and operational levels. Four problems are formulated: *room planning*, *teaching periods*, *assignment*, and *quality recovering*. Solution methods are provided for the two problems *room planning* and *teaching periods* that estimate the required resources. This output is used to create the timetable in the *assignment problem*, where a solution method based on a matheuristic is proposed. When assignments from the *course assignment problem* are disrupted, recovery is possible using the *quality recovering* method. This thesis shows how multi-objective optimization provides a better insight into the problems, and highlights trade-offs, allowing the planner to make a better decision based on solid foundations.

All of the proposed methods are based on MIP. In the more constrained recovering problems, where most assignments are fixed, optimal solutions can be quickly generated. However, the less-constrained assignment and strategic problems are still hard to solve optimally within a reasonable time. The performance of MIP solvers has improved significantly over the last decade, and ongoing improvements will directly influence the performance of the computations presented in this thesis.

The CB-CTT formulation provides the underlying model for all computational experiments. This formulation has been most widely-studied in the literature, which therefore makes it easy for other researchers to continue to develop these methods.

The use of a mathematical model makes the methods generic, and the timetabling formulation can take other soft and hard constraints into account without having to change the solution method. In practice, it is better to use the same solution method and underlying mathematical model in all steps of the timetabling process, in order to ensure that the model used in planning is the same as the one used to make strategic decisions. Decision support for these complex decision problems is essential if universities are to avoid wasting resources on poor planning (and instead use it for research and education).

5.2 Future Research

The proposed methods can still be improved, as finding optimal solutions based on MIP is difficult. With respect to the *assignment problem* the proposed matheuristics continue to be outperformed by metaheuristics and there is clearly potential for more research. To introduce variety into the chosen variables, the choice of neighborhoods is important, and finding better neighborhoods would be interesting. Predicting which neighborhoods offer improved solutions could be done adaptively using machine learning, which also would be a step towards making the method more general applicable. Strategic problems proved particularly difficult, given the large MIP gap between the upper and lower bound. It could be interesting to find a way to approximate soft constraints in order to be able to quickly estimate the consequence of removing, or adding, rooms and timeslots.

Timetabling is always based on limited information. Rooms are built without knowing what courses will offered in the future, and courses are assigned to timeslots and rooms without knowing what disruptions will occur. Incorporating these stochastic elements

can create a better understanding of the impact of these uncertainties. University course timetabling has been studied in the operations research community for over 30 years, but many universities continue to do their timetabling manually. One reason for this is that senior managers fail to appreciate the benefit of optimization. Universities, like every other organization, prefer to do things the way they have always done them. To incorporate optimization methods, universities need to invest in software and make changes to procedures which ensure that all of the necessary data is incorporated into the new system. This is not easy for managers, who need to be convinced of the usefulness of the exercise. Many will not consider that better solutions to the assignment problem is worth a big investment. As this thesis has highlighted, managers care most about expensive resources. Finding areas where resources are underutilized offers an opportunity to financially quantify improvements and create a convincing business case. Furthermore, the tactical problems described in Chapter 4 have not received much attention, and more research into how soft constraints interact and what are the best trade-offs will help to clarify how a good timetable looks like.

Another problem is that we can only work with a partial representation of reality when applying optimization. Every member of staff, and each student is different and cares more (or less) about the violation of soft constraints. One model will never be a complete representation of reality and it is, therefore, important that rather than focus on building software to generate the perfect timetable, we should support the planner in creating the perfect timetable. Consequently, there is a need to focus on providing the planner with insights and choices, leaving him or her to make the final decision. Multi-objective optimization is one way to achieve this goal, as the planner can make a decision based on information that goes beyond what is represented in the model. A related problem is how to present options to the planner, and provide an overview of the different solutions. Investigating interactions between decision-makers and software has been studied within the artificial intelligence community, where it is known as *human-computer symbiosis*.

The success of the first and second international timetabling competitions resulted in a lot of attention being given to solving the course assignment problem. To encourage researchers to take a broader perspective, it is likely that further progress could be made by launching a third timetabling competition, incorporating the following ideas:

- The formulation of a strategic problem, together with data instances.
- The formulation of a quality recovering problem, together with data instances.
- A multiple-stage competition, where the first stage is to determine resources, the second is to create a timetable, and the third is to recover from disruptions. A stochastic element could be introduced by limiting information about the next step.
- An advanced timetabling competition, inspired by advanced chess[1]. Here, the idea is that humans plus optimization software competes to create the best timetable in a very dynamic environment, in order to encourage intuitive user interfaces.

This thesis, which offers some new perspectives and publicly-available codebase, will hopefully encourage more research in these directions.

[1] See http://en.wikipedia.org/wiki/Advanced_Chess

Bibliography

Salwani Abdullah and Hamza Turabieh. On the use of multi neighbourhood structures within a tabu-based memetic approach to university timetabling problems. *information sciences*, 191:146–168, 2012.

Ramon Alvarez-Valdes, Enric Crespo, and Jose M. Tamarit. Assigning students to course sections using tabu search. *Annals of Operations Research*, 96(1-4):1–16, 2000. ISSN 0254-5330.

V. Bardadym. Computer-aided school and university timetabling: The new wave. In Edmund Burke and Peter Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 22–45. Springer Berlin / Heidelberg, 1996.

Roman Barták, Tomáš Müller, and Hana Rudová. A new approach to modeling and solving minimal perturbation problems. In *International Workshop on Constraint Solving and Constraint Logic Programming*, pages 233–249. Springer, 2003.

Timo Berthold. Primal heuristics for mixed integer programs. Master's thesis, Technischen Universitat Berlin, 2006.

Andrea Bettinelli, Valentina Cacchiani, Roberto Roberti, and Paolo Toth. An overview of curriculum-based course timetabling. *TOP*, pages 1–37, 2015.

Camille Beyrouthy, Edmund K Burke, Dario Landa-Silva, Barry McCollum, Paul McMullan, and Andrew J Parkes. Improving the room-size profiles of university teaching space. In *Dagstuhl Seminar on" Cutting, Packing, Layout and Space Allocation," March*. Citeseer, 2007.

Camille Beyrouthy, Edmund K Burke, Dario Landa-Silva, Barry McCollum, Paul McMullan, and Andrew J Parkes. Towards improving the utilization of university teaching space. *Journal of the Operational Research Society*, 60(1):130–143, 2009.

Camille Beyrouthy, Edmund K Burke, Barry McCollum, Paul McMullan, and Andrew J Parkes. University space planning and space-type profiles. *Journal of Scheduling*, 13 (4):363–374, 2010.

Robert E. Bixby. *Optimization Stories*, volume Extra of *21st International Symposium on Mathematical Programming Berlin*, chapter A Brief History of Linear and Mixed-Integer

Programming Computation, pages 107–121. Journal der Deutschen Mathematiker-Vereinigung, August 2012.

A. Bonutti, F. De Cesco, L. Di Gaspero, and A. Schaerf. Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results. *Annals of Operations Research*, 194(1):59–70, April 2012. ISSN 0254-5330.

Edmund Burke, Kirk Jackson, Jeffrey H. Kingston, and Rupert Weare. Automated university timetabling: The state of the art. *The computer journal*, 40(9):565–571, 1997.

Edmund K Burke, Jakub Marecek, Andrew J Parkes, and Hana Rudová. On a clique-based integer programming formulation of vertex colouring with applications in course timetabling. Technical report, Technical Report NOTTCSTR-2007-10, The University of Nottingham, 2007.

Edmund K. Burke, Jakub Mareček, Andrew J. Parkes, and Hana Rudová. Penalising patterns in timetables: Novel integer programming formulations. In Jörg Kalcsics and Stefan Nickel, editors, *Operations Research Proceedings 2007*, volume 2007 of *Operations Research Proceedings*, pages 409–414. Springer Berlin Heidelberg, 2008a. ISBN 978-3-540-77903-2. 10.1007/978-3-540-77903-2_63.

Edmund K. Burke, Jakub Mareček, Andrew J. Parkes, and Hana Rudová. A supernodal formulation of vertex colouring with application in course timetabling. *Annals of Operations Research*, 179(1):105–130, 2010. ISSN 0254-5330.

E.K. Burke, P. De Causmaecker, G.V. Berghe, and H. Van Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7:441–499, 2004. ISSN 1094-6136. 10.1023/B:JOSH.0000046076.75950.0b.

EK Burke, J Marecek, AJ Parkes, and H Rudová. Uses and abuses of mip in course timetabling. In *Poster at the Workshop on Mixed Integer Programming, MIP2007, Montréal*, 2008b.

V. Cacchiani, A. Caprara, R. Roberti, and P. Toth. A new lower bound for curriculum-based course timetabling. *Computers & Operations Research*, 40(10):2466 – 2477, 2013. ISSN 0305-0548.

Michael W Carter. A comprehensive course timetabling and student scheduling system at the university of waterloo. In *Practice and theory of automated timetabling III*, pages 64–82. Springer, 2001.

M.W. Carter. A survey of practical applications of examination timetabling algorithms. *Operations Research*, 34(2):193 – 202, 1986. ISSN 0030364X.

Emilie Danna, Edward Rothberg, and Claude Le Pape. Exploring relaxation induced neighborhoods to improve mip solutions. *Mathematical Programming*, 102(1):71–90, 2005.

Luca Di Gaspero and Andrea Schaerf. Neighborhood portfolio approach for local search applied to timetabling problems. *Journal of Mathematical Modelling and Algorithms*, 5:65–89, 2006. ISSN 1570-1166. 10.1007/s10852-005-9032-z.

Luca Di Gaspero, Barry McCollum, and Andrea Schaerf. The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3). Technical report, School of Electronics, Electrical Engineering and Computer Science, Queenes University SARC Building, Belfast, United Kingdom, 2007.

M. Dimopoulou and P. Miliotis. Implementation of a university course and examination timetabling system. *European Journal of Operational Research*, 130(1):202 – 213, 2001. ISSN 0377-2217.

Maria Dimopoulou and Panagiotis Miliotis. An automated university course timetabling system developed in a distributed environment: A case study. *European Journal of Operational Research*, 153(1):136–147, 2004.

W. Erben and J. Keppler. A genetic algorithm solving a weekly course-timetabling problem. In Edmund Burke and Peter Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 198–211. Springer Berlin / Heidelberg, 1996.

Jacques A Ferland and Serge Roy. Timetabling problem for university as assignment of activities to resources. *Computers & operations research*, 12(2):207–218, 1985.

Matteo Fischetti, Fred Glover, and Andrea Lodi. The feasibility pump. *Mathematical Programming*, 104(1):91–104, 2005.

Perry Fizzano and Steven Swanson. Scheduling classes on a college campus. *Computational optimization and applications*, 16(3):279–294, 2000.

C Roger Glassey and Michael Mizrach. A decision support system for assigning classes to rooms. *Interfaces*, 16(5):92–100, 1986.

Fred Glover and Manuel Laguna. *Tabu Search*. Springer, 2013.

Yacov Y Haimes, LS Ladson, and David A Wismer. Bicriterion formulation of problems of integrated system identification and system optimization, 1971.

Jin-Kao Hao and Una Benlic. Lower bounds for the ITC-2007 curriculum-based course timetabling problem. *European Journal of Operational Research*, 212(3):464 – 472, 2011. ISSN 0377-2217.

Alain Hertz. Tabu search for large scale timetabling problems. *European journal of operational research*, 54(1):39–47, 1991.

Informs. Science of better website, 2017. URL `http://www.scienceofbetter.org/`.

G. Kendall, S. Knust, C.C. Ribeiro, and S. Urrutia. Scheduling in sports: An annotated bibliography. *Computers & Operations Research*, 37(1):1 – 19, 2010. ISSN 0305-0548.

A. Kiefer, R. Hartl, and A. Schnell. Adaptive large neighborhood search for the curriculum-based course timetabling problem. Technical report, University of Vienna, 2014.

J. Kingston. Resource assignment in high school timetabling. *Annals of Operations Research*, pages 1–14, 2010. ISSN 0254-5330. 10.1007/s10479-010-0695-0.

Jeffrey H. Kingston. Educational timetabling. In A. Sima Uyar, Ender Ozcan, and Neil Urquhart, editors, *Automated Scheduling and Planning*, volume 505 of *Studies in Computational Intelligence*, pages 91–108. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-39303-7.

Simon Kristiansen and Thomas R. Stidsen. A comprehensive study of educational timetabling - a survey. Technical Report 8, 2013, DTU Management Engineering, Technical University of Denmark, November 2013.

PeterJ.M. Laarhoven and EmileH.L. Aarts. Simulated annealing. In *Simulated Annealing: Theory and Applications*, volume 37 of *Mathematics and Its Applications*, pages 7–15. Springer Netherlands, 1987. ISBN 978-90-481-8438-5.

G. Lach and M. Lübbecke. Optimal university course timetables and the partial transversal polytope. In Catherine McGeoch, editor, *Experimental Algorithms*, volume 5038 of *Lecture Notes in Computer Science*, pages 235–248. Springer Berlin / Heidelberg, 2008.

G. Lach and M. Lübbecke. Curriculum based course timetabling: new solutions to udine benchmark instances. *Annals of Operations Research*, 194:255–272, 2012. ISSN 0254-5330.

Ailsa H Land and Alison G Doig. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, pages 497–520, 1960.

Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, 1992.

R. Lewis, B. Paechter, and B. McCollum. Post enrolment based course timetabling: A description of the problem model used for track two of the second international timetabling competition. Cardiff Accounting and Finance Working Papers A2007/3, Cardiff University, Cardiff Business School, Accounting and Finance Section, 2007.

Rhydian Lewis. A survey of metaheuristic-based techniques for university timetabling problems. *OR Spectrum*, 30:167–190, 2008. ISSN 0171-6468. 10.1007/s00291-007-0097-0.

Andrea Lodi. The heuristic (dark) side of mip solvers. In El-Ghazali Talbi, editor, *Hybrid Metaheuristics*, volume 434 of *Studies in Computational Intelligence*, pages 273–284. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-30670-9.

Leo Lopes and Kate Smith-Miles. Pitfalls in instance generation for udine timetabling. In *International Conference on Learning and Intelligent Optimization*, pages 299–302. Springer, 2010.

Leo Lopes and Kate Smith-Miles. Generating applicable synthetic instances for branch problems. *Operations Research*, 61(3):563–577, 2013.

Clarence H. Martin. Ohio university's college of business uses integer programming to schedule classes. *Interfaces*, 34(6):460–465, November 2004.

Barry McCollum. A perspective on bridging the gap between theory and practice in university timetabling. In EdmundK. Burke and Hana Rudova, editors, *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pages 3–23. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-77344-3.

Barry McCollum, Andrea Schaerf, Ben Paechter, Paul McMullan, Rhyd Lewis, Andrew J. Parkes, Luca Di Gaspero, Rong Qu, and Edmund K. Burke. Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, 22(1):120–130, 2010.

Amnon Meisels and Andrea Schaerf. Modelling and solving employee timetabling problems. *Annals of Mathematics and Artificial Intelligence*, 39(1-2):41–59, 2003.

S. A. MirHassani and F. Habibi. Solution approaches to the course timetabling problem. *ARTIFICIAL INTELLIGENCE REVIEW*, 39(2):133–149, 2013. ISSN 02692821, 15737462. doi: 10.1007/s10462-011-9262-6.

Moritz Mühlenthaler and Rolf Wanka. Fairness in academic course timetabling. *Annals of Operations Research*, 239(1):171–188, 2016.

T. Müller. Itc2007 solver description: a hybrid approach. *Annals of Operations Research*, 172:429–446, 2009. ISSN 0254-5330.

Tomáš Müller and Keith Murray. Comprehensive approach to student sectioning. *Annals of Operations Research*, 181:249–269, 2010. ISSN 0254-5330.

Tomáš Muller, Hana Rudová, and Roman Barták. Minimal perturbation problem in course timetabling. In Edmund Burke and Michael Trick, editors, *Practice and Theory of Automated Timetabling V*, volume 3616 of *Lecture Notes in Computer Science*, pages 126–146. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-30705-1.

Keith Murray, Tomáš Muller, and Hana Rudová. Modeling and solution of a complex university course timetabling problem. In Edmund Burke and Hana Rudová, editors, *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pages 189–209. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-77344-3.

Metaheuristics Network. International timetabling competition, 2002.

Ben Paechter, Andrew Cumming, Michael G Norman, and Henri Luchian. Extensions to a memetic timetabling system. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 251–265. Springer, 1995.

Antony E Phillips. *Mathematical Programming-based Model and Methods for University Course Timetabling*. PhD thesis, University of Auckland, 2015.

Antony E. Phillips, Hamish Waterer, Matthias Ehrgott, and David M. Ryan. Integer programming methods for large-scale practical classroom assignment problems. *Computers & Operations Research*, 53(0):42 – 53, 2015. ISSN 0305-0548. doi: http://dx.doi.org/10.1016/j.cor.2014.07.012. URL `http://www.sciencedirect.com/science/article/pii/S0305054814001956`.

Antony E Phillips, Cameron G Walker, Matthias Ehrgott, and David M Ryan. Integer programming for minimal perturbation problems in university course timetabling. *Annals of Operations Research*, pages 1–22, 2016.

Hana Rudová, Tomáš Muller, and Keith Murray. Complex university course timetabling. *Journal of Scheduling*, 14(2):187–207, 2011. ISSN 1094-6136.

J Dario Landa Silva, Edmund K Burke, and Sanja Petrovic. An introduction to multiobjective metaheuristics for scheduling and timetabling. In *Metaheuristics for multiobjective optimisation*, pages 91–129. Springer, 2004.

Matias Sørensen and Thomas R. Stidsen. High school timetabling: Modeling and solving a large number of cases in denmark. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 359–364. SINTEF, 2012.

Eric Taillard. Benchmarks for basic scheduling problems. *european journal of operational research*, 64(2):278–285, 1993.

A. Wren. Scheduling, timetabling and rostering a special relationship? In Edmund Burke and Peter Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 46–75. Springer Berlin / Heidelberg, 1996.

Part II

Scientific Papers

Chapter 6

A Fix-and-Optimize Matheuristic for Curriculum-Based Course Timetabling

Michael Lindahl · Matias Sørensen · Thomas R. Stidsen

Abstract University course timetabling covers the task of assigning rooms and time periods to courses while ensuring a minimum violation of soft constraints that define the quality of the timetable. These soft constraints can have attributes that make it difficult for mixed-integer programming solvers to find good solutions fast enough to be used in a practical setting. Therefore, metaheuristics have dominated this area despite the fact that mixed-integer programming solvers have improved tremendously over the last decade. This paper presents a matheuristic where the MIP-solver is guided to find good feasible solutions faster. This makes the matheuristic applicable in practical settings, where mixed-integer programming solvers do not perform well. To the best of our knowledge this is the first matheuristic presented for the University Course Timetabling problem.

The matheuristic works as a large neighborhood search where the MIP solver is used to explore a part of the solution space in each iteration. The matheuristic uses problem specific knowledge to fix a number of variables and create smaller problems for the solver to work on, and thereby iteratively improves the solution. Thus we are able to solve very large instances and retrieve good solutions within reasonable time limits. The presented framework is easily extendable due to the flex-

ibility of modeling with MIPs; new constraints and objectives can be added without the need to alter the algorithm itself. At the same time, the matheuristic will benefit from future improvements of MIP solvers.

The matheuristic is benchmarked on instances from the literature and the 2nd International Timetabling Competition (ITC2007). Our algorithm gives better solutions than running a state-of-the-art MIP solver directly on the model, especially on larger and more constrained instances. Compared to the winner of ITC2007, the matheuristic performs better. However, the most recent state-of-the-art metaheuristics outperform the matheuristic.

Keywords Matheuristics · Integer programming · University course timetabling

6.1 Introduction

University course timetabling is the problem of assigning courses to rooms and time periods. This should be done without violating a number of hard constraints that would make the timetable infeasible. For example, a teacher is only able to teach one class in a certain time period. Furthermore, a number of soft constraints should be obeyed as much as possible, because violating them would result in a timetable with undesired features. An undesired feature could e.g. be lectures of one course planned in different rooms during the week. The generating of high quality timetables automatically leads to better timetables in a production setting, and the automatization can help planners make timetables faster as they often have tight deadlines.

Timetabling differs between universities according to traditions and how their educations are structured. Many universities have curricula which consist of a number of courses taken by a group of students in a specific semester. This problem formulation is called the Curriculum-based Course Timetabling (CB-CTT). A formal description of this problem was made for the Second International Timetabling Competition in 2007 by Di Gaspero et al. [2007] together with 21 benchmark instances from the University of Udine. This allowed researchers to compare their results on the same instances with roughly the same computational resources. The rules of the competition disallowed the use of external solvers which meant that all contestants used metaheuristics. A website has been made to keep track of the best known solutions and the best known bounds. The website is maintained by A. Bonutti, L. Gaspero and A. Schaerf and can be found at `http://tabu.diegm.uniud.it/ctt/`.

The purpose of this paper is to combine mixed-integer programming and heuristics to find good feasible solutions fast, taking advantage of the large improvements in mixed-integer programming solvers that have happened over the last decade as shown in Bixby [2012]. This combination is commonly known as *matheuristics*. The matheuristic developed in this paper works as a large neighborhood search where the MIP solver is used to explore a part of the solution space in each iteration. The heuristic uses problem specific knowledge to fix a number of variables and create smaller problems for the solver to work on and thereby iteratively improves the solution. Thus, we are able to solve very large

instances and retrieve good solutions within reasonable time limits. The presented framework is easily extendable due to the flexibility of modeling with MIPs; new constraints and objectives can be added without the need to alter the algorithm itself. At the same time, the matheuristic will benefit from future improvements in MIP solvers.

The paper is organized as follows: In section 6.2 we describe previous work. In section 6.3 the Curriculum-based Course Timetabling problem is defined together with the MIP formulation used. In section 6.4 the matheuristic is described. The computational results of the algorithm are shown in section 6.5. Finally, the conclusion and outlook of future directions are presented in section 6.6.

6.2 Previous work

The CB-CTT problem has received a lot of attention in the literature as the de-facto benchmarking problem within university timetabling. The winning heuristic of the original competition is described in Müller [2009]. More recently, especially two heuristics have shown to perform well on the problem, see Abdullah and Turabieh [2012] and Kiefer et al. [2014]. As the algorithm presented in this paper is based on a MIP solver, we will focus on other MIP-based approaches in this section.

Integer programming has been applied to CB-CTT, but the models have proved difficult to solve due to the characteristics of the problem described by Burke et al. [2008]. In Hao and Benlic [2011] they successfully generate lower bounds by using MIP and a partition-based approach based on the "divide and conquer principle", and Cacchiani et al. [2013] split the objective function into two parts formulated as MIPs that are solved separately by using a column generation procedure. In Lach and Lübbecke [2012] the problem is separated in two stages (two MIP models) that are exact with respect to the hard constraints when solved in sequence. They generate lower bounds and are also able to generate good feasible solutions. For a complete overview of the different methods applied to CB-CTT we recommend the overview by Bettinelli et al. [2015]. Hybridizing exact methods with heuristics is not a new idea. One of the earliest examples is the corridor method by Sniedovich and Voß [2006], where a corridor is made around the current solution to create a smaller search space. Within MIP Danna et al. [2005] proposed a relaxation induced neighborhood search (RINS) by using the current incumbent together with the linear relaxation to search for improving solutions. New heuristics have been proposed, for example the local branching by Fischetti and Lodi [2003] who use the concept of hamming distance to create a branching based on the current solution. Within timetabling Avella et al. [2007] applie a local search algorithm on a high-school timetabling problem and use a MIP solver to explore a very large neighborhood.

6.3 Curriculum-based Course Timetabling

In curriculum-based course timetabling (CB-CTT) the purpose is to assign a number of lectures to a time period and a room. To be able to compare results to the current state-of-the-art we use the formulation from the Second International Timetabling Competition ITC-2007 stated in Di Gaspero et al. [2007].

A set of courses is given, and each course consists of a number of lectures that should be planned. A lecture should be assigned to a time period and a room without causing any conflicts. A set of time periods is given, and each one is associated with a day and a time, and two lectures of the same course can not take place at the same time or it will result in a conflict. A set of rooms is given, and only one lecture can take place in the same room in a time period, or it will result in a conflict. Each course also has a teacher assigned, and it will result in a conflict if a teacher has two lectures in the same time period. Furthermore, a set of curricula is also given. A curriculum consists of a set of courses, and courses from the same curriculum will conflict if they are planned in the same time period.

All lectures should be planned without conflicts, and the objective is to create a timetable that minimizes the violation of a number of soft constraints. The soft constraints define some attributes that we would like to avoid. Each one is associated with a number of penalty points, and the goal is therefore to minimize the total sum of penalty points for the timetable. The following four soft constraints are defined:

RoomCapacity Each room has a capacity and each course has a number of students who attend the course. If a course is planned in a room with less capacity than the number of students attending the course, *one penalty point* is given for each student exceeding this number.

RoomStability A course consists of several lectures and it is desired that all of these are assigned to the same room. *one penalty point* is given for each additional room used.

MinimumWorkingDays To distribute the workload of a course throughout the week it is desired to spread the lectures out on several days. Each course has a number of minimum working days which should be respected. For each day below *five penalty points* are given.

CurriculumCompactness To avoid that students have idle time periods in the timetable it is desired that curricula are planned consecutively. If a course of a curriculum is planned in a time period where there is not another course from the same curriculum in the previous or in following time period, *two penalty points* are given.

6.3.1 Mixed Integer Programming Model

The proposed matheuristic works on top of a MIP model. Different models for the CB-CTT have been proposed in the literature and we refer to Bettinelli et al. [2015] for an overview of these. We use the MIP model proposed in Lach and Lübbecke [2012]. This model has shown good results on the ITC2007 benchmark instances, and we believe that this is currently the best model when the goal is to create feasible solutions. The approach consists of two stages solved sequentially. The first stage assigns time periods to the lectures, and the second stage then assigns the rooms. The decomposition is exact with respect to the hard constraints, meaning that no feasible solutions are lost.

The following sets are defined:

$\mathcal{C}$: Set of courses

$\mathcal{CU}$**:** Set of curricula

$\mathcal{P}$**:** Set of timeslots across the week

$\mathcal{D}$**:** Set of days of the week

$\mathcal{R}$**:** Set of rooms

$\mathcal{T}$**:** Set of teachers

The following parameters are defined:

$l(c)$**:** The number of lectures for course $c \in \mathcal{C}$

$mnd(c)$**:** The minimum working days for course $c \in \mathcal{C}$

$dem(c)$**:** The demand for course $c \in \mathcal{C}$ i.e. the number of students.

$cap(r)$**:** The capacity of room $r \in \mathcal{R}$

$\mathcal{C}(t)$ Set of courses where the teacher is $t \in \mathcal{T}$

$\mathcal{C}(cu)$ Set of courses included in curriculum $c \in \mathcal{CU}$

Furthermore, the following helper sets are used to formulate the model:

$\mathcal{C}_{\geq s} = \{c \in \mathcal{C} : dem(c) \geq s\}$, set of courses with a demand larger than or equal to $s \in \mathcal{S}$

$\mathcal{R}_{\geq s} = \{r \in \mathcal{R} : cap(r) \geq s\}$, set of rooms with capacity larger than or equal to $s \in \mathcal{S}$

Stage I

The first stage determines at what time periods each lecture should be taught. It does not assign any rooms, but keeps track of the available rooms to ensure that a feasible room assignment exists and calculates the violation of the *RoomCapacity* constraint. The Stage I therefore considers all soft constraints except for *RoomStability*. The decision variable is defined as,

$$x_{c,p} = \begin{cases} 1 & \text{if course } c \in \mathcal{C} \text{ is planned at period } p \in \mathcal{P} \\ 0 & \text{otherwise} \end{cases}$$

The following auxiliary variables are also needed: To keep track of the violation of *RoomCapacity*, the variable $y_{s,c,p}$ determines if course $c \in \mathcal{C}$ in period $p \in \mathcal{P}$ is planned in a room of size $s \in \mathcal{S}$ or smaller. The variable $z_{c,d}$ is equal to one if course $c \in \mathcal{C}$ is planned on day $d \in \mathcal{D}$. The variable w_c then counts the number of violations of the *MinimumWorkingDays* constraint for course $c \in \mathcal{C}$. The variable $r_{cu,p}$ is one if a course from curriculum $cu \in \mathcal{CU}$ is planned in period $p \in \mathcal{P}$. The violation of the *CurriculumCompactness* is then calculated by $v_{cu,p}$ that determines if there is an isolated lecture from curriculum $cu \in \mathcal{CU}$ in period $p \in \mathcal{P}$. The full model for stage I is shown in Model 1.

The objective (1a) calculates the violation of the three soft constraints. The constraints of the MIP are described in the following:

$$\min \quad \sum_{p\in\mathcal{P},s\in\mathcal{S},c\in\mathcal{C}_{\geq s}} obj_{s,c,p}\cdot y_{s,c,p} + \sum_{c\in\mathcal{C}} 5\cdot w_c + \sum_{cu\in\mathcal{CU},p\in\mathcal{P}} 2\cdot v_{cu,p} \tag{1a}$$

$$\text{s.t.} \quad \sum_{p\in\mathcal{P}} x_{c,p} = L(c) \qquad \forall c\in\mathcal{C} \tag{1b}$$

$$\sum_{c\in\mathcal{C}} x_{c,p} \leq |\mathcal{R}| \qquad \forall p\in\mathcal{P} \tag{1c}$$

$$x_{c,p} - y_{s,c,p} \geq 0 \qquad \forall c\in\mathcal{C}, p\in\mathcal{P}, s\in\mathcal{S} \tag{1d}$$

$$\sum_{c\in\mathcal{C}_{\geq s}} (x_{c,p} - y_{s,c,p}) \leq |\mathcal{R}_{\geq s}| \qquad \forall s\in\mathcal{S}, p\in\mathcal{P} \tag{1e}$$

$$\sum_{p\in\mathcal{P}} x_{c,p} - z_{c,d} \geq 0 \qquad \forall c\in\mathcal{C}, d\in\mathcal{D} \tag{1f}$$

$$\sum_{d\in\mathcal{D}} z_{c,d} + w_c \geq mnd(c) \qquad \forall c\in\mathcal{C} \tag{1g}$$

$$\sum_{c\in\mathcal{C}(cu)} x_{c,p} - r_{cu,p} = 0 \qquad \forall cu\in\mathcal{CU}, p\in\mathcal{P} \tag{1h}$$

$$-r_{cu,p-1} + r_{cu,p} - r_{cu,p+1} - v_{cu,p} \leq 0 \qquad \forall cu\in\mathcal{CU}, p\in\mathcal{P} \tag{1i}$$

$$\sum_{c\in\mathcal{C}(t)} x_{c,p} \leq 1 \qquad \forall t\in\mathcal{T}, p\in\mathcal{P} \tag{1j}$$

$$x_{c,p}\in\mathbb{B} \qquad \forall c\in\mathcal{C}, p\in\mathcal{P} \tag{1k}$$

$$y_{s,c,p}\in\mathbb{B} \qquad \forall s\in\mathcal{S}, c\in C_{\geq s}, p\in\mathcal{P} \tag{1l}$$

$$w_c\in\mathbb{Z}_+ \qquad \forall c\in\mathcal{C} \tag{1m}$$

$$z_{c,d}\in\mathbb{B} \qquad \forall c\in\mathcal{C}, d\in\mathcal{D} \tag{1n}$$

$$v_{cu,p}\in\mathbb{B} \qquad \forall cu\in\mathcal{CU}, p\in\mathcal{P} \tag{1o}$$

$$r_{cu,p}\in\mathbb{B} \qquad \forall cu\in\mathcal{CU}, p\in\mathcal{P} \tag{1p}$$

Model 1: The MIP model for the first stage.

Constraints

(1b) – A course $c\in\mathcal{C}$ should be assigned exactly L_c lectures.

(1c) – At given period $p\in\mathcal{P}$ it is not possible to assign more courses than available rooms.

(1d) – If a room size is assigned to a course, the course should be assigned to that time period.

(1e) – Ensures that we do not assign more courses to a certain room size than rooms available of that size.

(1f) – Calculates if a course $c \in \mathcal{C}$ is planned on day $d \in \mathcal{D}$

(1g) – Calculates the violation of the *MinimumWorkingDay* constraint.

(1h) – Calculates if a curriculum is planned in a time period and ensures that only one course from the same curriculum is planned.

(1i) – Calculates the *CurriculumCompactness* violation.

(1j) – Ensures that only one course with the same teacher is planned at the same period.

Stage II

The Stage II model assigns a room to each lecture based on the solution of Stage I while minimizing *RoomStability*. The decision variable is the following:

$$u_{c,p,r} = \begin{cases} 1 & \text{if course } c \in \mathcal{C} \text{ is planned in room } r \in \mathcal{R} \text{ at period } p \in \mathcal{P} \\ 0 & \text{otherwise} \end{cases}$$

The assignments of times is given from the previous stage as $x^*_{c,p}$. If a course is not assigned to a time period, no room should be assigned. The violation of room capacity is given from the variables $y^*_{s,c,p}$. This means that if a lecture has been assigned to a smaller room in Stage I this should also be the case in the second stage.

Given a course $c \in \mathcal{C}$, a time period $p \in \mathcal{P}$ and a room $r \in \mathcal{R}$ then the assignment is invalid, i.e. $u_{c,p,r} = 0$, if one of these three conditions is satisfied:

- If a course c is not assigned to a time period p then no room should be assigned.
  - $x^*_{c,p} = 0$
- If there is no violation of the capacity in Stage I then there should not be a violation in Stage II.
  - $y^*_{s,c,p} = 0$ and $dem(c) > cap(r)$
- If the capacity is exceeded in Stage I, then the same violation should occur in Stage II.
  - $y^*_{s,c,p} = 1$
  - $dem(c) \leq cap(r)$
  - $cap(r) = \max_{\hat{r} \in \mathcal{R}} \{cap(\hat{r}) : cap(\hat{r}) < dem(c))\}$

Furthermore, the variable $y_{c,r}$ states whether course $c \in \mathcal{C}$ takes place in room $r \in \mathcal{R}$ at least once. This is used to calculate the violation of the *RoomStability* constraint. The full model of Stage II is shown in Model 2.

The objective (2a) is the violation of the *RoomStability* objective. The constraints are the following:

$$\min \quad \sum_{c \in \mathcal{C}, r \in \mathcal{R}} y_{c,r} \tag{2a}$$

$$\text{s. t.} \quad \sum_{p \in \mathcal{P}} u_{c,p,r} - |\mathcal{P}| \cdot y_{c,r} \leq 0 \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{2b}$$

$$\sum_{r \in \mathcal{R}} u_{c,p,r} = 1 \quad \forall c \in \mathcal{C}, p \in \mathcal{P} : x^*_{c,p} = 1 \tag{2c}$$

$$\sum_{c \in \mathcal{C}, p \in \mathcal{P}} u_{c,p,r} \leq 1 \quad \forall r \in \mathcal{R} \tag{2d}$$

$$y_{c,r} \in \mathbb{B} \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{2e}$$

$$u_{c,p,r} \in \mathbb{B} \quad \forall c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R} \tag{2f}$$

Model 2: The MIP model for the second stage.

Constraints

(2b) – Calculates the violation of the *RoomCapacity* constraint.

(2c) – Ensures that all lectures get assigned to a room.

(2d) – Ensures that no more than one lecture is assigned to the same room.

6.4 Fix-and-Optimize Matheuristic

A natural way of finding good solutions to a problem is to iteratively improve a bad solution. This is done by creating a neighborhood that can be explored around the solution, like proposed in the Large Neighborhood Search (LNS) algorithm (Shaw [1998]). An example of a matheuristic is the corridor method proposed by Sniedovich and Voß [2006]. In the corridor method an exact method that can solve small instances of a problem, but which is not applicable to large instances, is used. A corridor around the current solution is created, and this results in a smaller neighborhood which can then be explored by using the exact method. Our matheuristic build on a similar approach, as we have a mixed-integer model that can solve small instances to optimality, but struggles with large instances. In each iteration the MIP solver is used to explore a large neighborhood defined by fixing a subset of the variables. This results in a new MIP that we call the *subproblem*, and the solver is then used as a black-box to search for improving solutions within this. This is the *fix-and-optimize* aspect of the matheuristic. This also implies that if the original model is changed, e.g. a constraint is added or removed, the neighborhoods are not affected as the same model is used. This can sometimes be a problem in move-based heuristics where a new constraint can make certain moves obsolete as they depend too much on special characteristics of the solution.

An important aspect of implementing this algorithm is that the model is only built from scratch once by the MIP-solver. As the same model instance is used throughout the algorithm, and due to the way the fixing of the variables is defined, the solution found in the previous iteration of the algorithm is always feasible with regard to the fixing of the variables in the next iteration. For most MIP solvers, this means that the solve operation in each iteration is automatically warm-started from the previously found solution, which is a great benefit. For the MIP solver used in these experiments (Gurobi) this is certainly the case.

In Caserta and Voss [2010] they put metaheuristics in two classes. The first is the *model based heuristics* where a new solution is found by using a model. Our algorithm falls in the second category called *method based heuristic* where the underlying model is still the same, but the neighborhood is defined by how the MIP solver explores the neighborhood.

The proposed method is applied to Stage I of the problem as experiments have shown that the majority of the solution time is spent at this stage. The resulting solution is then handed to the Stage II model to find a solution for the entire problem.

The algorithm is described in Algorithm 6.4.1. A key part of the algorithm is to determine which variables to fix to create subproblems that are easier to solve than the full problem, while still leading to improving solutions. This part will be explained in Section 6.4.2.

Algorithm 6.4.1 Fix-and-Optimize Matheuristic

input:
    problem instance
    Set of neighborhoods N and initial sizes S
output:
    Solution
$x^*_{c,p} \leftarrow$ Create Initial solution
Fix all decision variables
while stopping criteria not met **do**
    Pick neighborhood $n \in N$
    $X(R)$ = NeighborhoodCreator(n) ▷ Find connected decision variables
    Unfix variables $X(R)$
    $x^*_{c,p} \leftarrow$ Optimize subproblem
    Update S_n ▷ Use feedback from solver to update parameters
end while
Solve StageII($x^*_{c,p}$)

6.4.1 Initial solution

Before decisions variables can be fixed, a feasible initial solution is needed. As shown in Lach and Lübbecke [2012] the problem can quickly be solved when the soft constraint *CurriculumCompactness* is removed. Moreover, the objective *MinimumWorkingDays* also introduces new auxiliary variables, and removing this makes the problem easier. Creating

a start solution is therefore done by removing those two objectives: *CurriculumCompactness* and *MinimumWorkingDays*. To avoid spending too much time in this step the solver is set to return the first found solution.

6.4.2 Neighborhoods

A neighborhood defines which part of the solution space should be explored in an iteration. As mentioned, this is done by choosing a subset of variables that should be unfixed thus allowing the values to be changed.

The neighborhood should choose variables where it is likely that a change will result in an improved solution. If the resulting subproblem is too constrained by the fixed variables, then no new solutions will be found. At the same time it should also be possible for the solver to find an improving solution which depends on how difficult the subproblem is to solve. If the neighborhood is too difficult to solve within the time limit of each iteration, no improvement will be made.

There is no exact way to predict how difficult a MIP model is to solve, but as mentioned by Vielma [2015] the size of the model has a high impact. A simple measure which also is easy to calculate, is the number of decision variables. We therefore define the *size* of a neighborhood as the number of unfixed decision variables. In each iteration we have a neighborhood size, $S \in \mathbb{Z}_+$, which will determine how many variables to pick.

It is also important to choose decision variables that are connected in the sense that if one of them changes value, the others are likely to change value as well. An example is that if an assignment of a course in a curriculum is moved to a new time period, it is likely to affect other courses in that same curriculum, as they may need to be assigned to a new time period to avoid a conflict.

We define a resource r as an entity associated to a set of decision variables. We define the variables associated with resource r as $X(r)$. A resource could for example be a course where the associated decision variables are the ones that affect that course. i.e. to a course $c' \in \mathbb{C}$ the corresponding decision variables are

$$X(c') = X_{c,p} : c = c', p \in \mathbb{P}$$

To pick resources that are connected we create a *score* function to measure how well a specific resource is connected to a set of other resources. A high number means that if a decision variable from the set of resources is changed, it will likely result in a change in the given resource.
The score of resource r related to the list of resources R is defined the following way:

$$Score(R, r) \in \mathbb{R}_0^+$$

The algorithm uses a greedy heuristic to find connected resources. It starts with a list of resources of the same type, for example all courses. It then picks the first at random and then iteratively adds more depending on which one has the largest score. This is repeated until the desired size is reached. The algorithm is shown in Algorithm 6.4.2.

Three different neighborhoods are defined. The purpose of this is to make sure that it is possible to reach different parts of the solution space. The neighborhoods are put into two categories, static and dynamic. The static neighborhoods look at the problem

Algorithm 6.4.2 NeighborhoodCreator

input: List of resources R, Neighborhood Size S
output: List of decision variables $\bar{X}$
$\bar{R} := \{Random_{r \in R}\}$ ▷ Pick a random resource
while $|X(\bar{R})| < S$ **do** ▷ Add more until desired size is reached
    $r = \arg\max_{r \in R} Score(\{\bar{R}, r\})$ ▷ Find resource with highest score
    $\bar{R} := \bar{R} \cup r$ ▷ Add resource
end while
return $X(\bar{R})$ ▷ Return corresponding decision variables

formulation to pick variables where the dynamic looks at the current solution too. The three neighborhoods are: *Curricula*, *Courses* and *Assignments*.

Curricula The curricula neighborhood is static and chooses the decision variables of courses that belong to a set of curricula. Looking at optimal solutions, the objective *CurriculumCompactness* is usually contributing to a large part of the penalty. The purpose of this neighborhood is to make sure that the parts of the solution space that can lead to lowering this are explored. The score is calculated by choosing curricula with overlapping courses as moving one lecture will influence both curricula. The resources are therefore curricula $\mathcal{CU}$ and the score function is the number of courses in common:

$$Score(cu', \mathcal{CU}') = |c \in cu'| : c \in \mathcal{CU}'$$

Courses This neighborhood finds courses that have similar numbers of students attending and is therefore static. This is to be able to swap courses that use similar rooms. The resource is courses $\mathcal{C}$ with the following score function that measures the difference from the average number of students:

$$Score(c', \mathcal{C}') = -\left| dem(c') - \frac{1}{|\mathcal{C}'|} \sum_{c \in \mathcal{C}'} dem(c) \right|$$

Assignments This dynamic neighborhood looks at the assignments in the current best solution and looks at how much the assignment violates the soft constraints . The purpose of this is to fix undesired assignments that result in lowering the objective. Each assignment is therefore associated with the amount of violation in which it results.

$$Score(y'_{s',c',p'}, Y) = obj_{s',c',p'} + 5 \cdot w'_c + \sum_{p,cu \in \mathcal{CU}: c \in \mathcal{CU}} 2 \cdot r_{cu,p}$$

6.4.3 Choosing neighborhood

In each iteration it needs to be decided which neighborhood to use. This is done in a random manner by selecting what neighborhood to use from a uniform distribution.

6.4.4 Adaptive Neighborhood Size

Finding the right size of the neighborhoods is important. We therefore continually adjust the neighborhood size S to ensure that we are solving neither too easy or too difficult subproblems.

A way to measure the hardness of a subproblem is to look how far from optimum we are when the time limit of an iteration is reached. This is done by looking at the relative gap between the lower bound and the incumbent for each subproblem. We define two parameters *MinGap* and *MaxGap* and aim at creating subproblems that are solved within this gap. If the gap is less than *MinGap* the neighborhood size of that neighborhood is increased, and in the same way decreases it if it is larger than *MaxGap*.

The increase or decrease of the neighborhood size is defined in terms of the parameter *Decay*. Because each neighborhood is different, we have a size S for each of the used neighborhoods. The adaptive algorithm for updating the sizes is shown in Algorithm 6.4.3.

Algorithm 6.4.3 UpdateNeighborhoodsize

```
1: if MipGap > MaxGap then                    ▷ Subproblem is too difficult
2:     Size *= 1 - Decay                      ▷ Decrease size
3: else if MipGap < MinGap then               ▷ Subproblem is too easy
4:     Size *= 1 + Decay                      ▷ Increase size
5: end if
```

Smoothing The amount of decision variables is not enough to predict how difficult a subproblem is. Therefore, a lot of fluctuation in the MIP Gap happens between iterations for the same number of decision variables. To handle this we smooth the MIP Gap by using *exponential smoothing*, which makes it less fluctuated by averaging the MIP Gap with its previous value. This is done in the following way: Let $MipGap_i$ be the gap in iteration i, the smoothed gap, denoted $\widetilde{MipGap}_i$, is then calculated in the following way, where α is the *smoothing factor*.

$$\begin{aligned} \widetilde{MipGap}_1 &= MipGap_1 \\ \widetilde{MipGap}_i &= \alpha \cdot MipGap_i + (1-\alpha) \cdot \widetilde{MipGap}_{i-1}, \quad i > 1 \end{aligned}$$

An example of how the size of the neighborhood adapts when using the MipGap is seen in Figure 6.1. It is seen that there is a lot of fluctuation in the MIPGap, but it is smoothed by the *exponential smoothing*.

6.5 Computational results

To evaluate the algorithm we use both the instances from the ITC2007 competition and the much larger instances from University of Erlangen. All data sets are available from `http://tabu.diegm.uniud.it/ctt/`. We will compare the algorithm to running the MIP solver directly on the CB-CTT model. As the goal is to provide a tool for practical

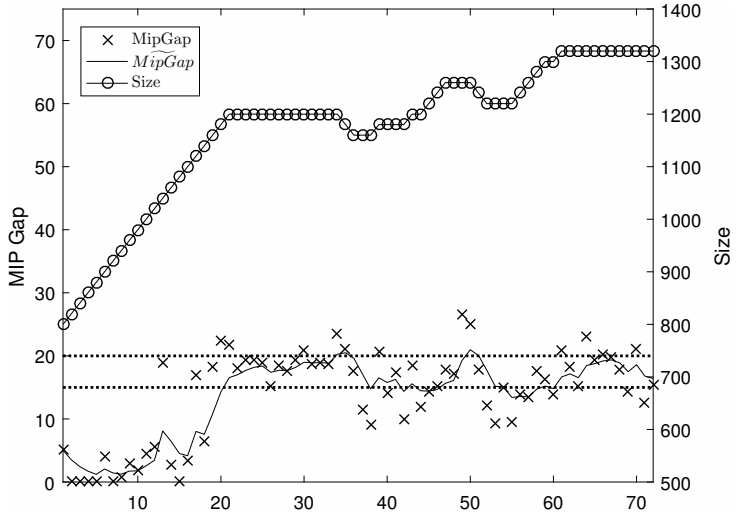


Figure 6.1: This figure illustrates the size of the neighborhood and the MipGap in each iteration. It can be seen that the size of the neighborhood is gradually increased until it is stabilized and then increased in the end again.

Neighborhood	Curricula	Courses	Assignments
Size	1000	2500	60%

Table 6.1: The initial sizes of the neighborhoods. Two of them are absolute number of decision variables and the other are a percentage of the total amount.

timetabling, we will compare to the state-of-the-art in metaheuristic, both the winner from the original ITC2007 competition as well as newer algorithms proposed in the literature.

All experiments were run on a 64 bit Windows machine with a 4.00GHz CPU and 32GB of memory. To solve the integer programs we use Gurobi 6.5.0 running with standard parameters except for `MIPFocus = 1` to tell the solver to focus on finding feasible solutions instead of proving optimality. The solver is only allowed to use one thread. The allowed time limit is determined by using the benchmark tool provided for the ITC2007 competition. This means that *one CPU time unit* corresponds to 260 seconds. We use the same fraction on time for Stage I and II as proposed by Lach and Lübbecke [2012], where approximately 75% of the time is used in Stage I. This means that Stage I is given 210 seconds and Stage II 50 seconds.

The time limit for each iteration in the matheuristic is set to *2 seconds*. For the adaptive part the following parameters are used: `GapMin` $= 15\%$, `GapMax` $= 20\%$, $\alpha = 0.3$ and `Decay` $= .02$. The initial sizes of the neighborhoods are shown in Table 6.1. These are either an absolute number of decision variables or as an percentage of the total amount in the given instance.

All results are shown as the average over 10 runs with different random seeds to the matheuristic and the MIP solver.

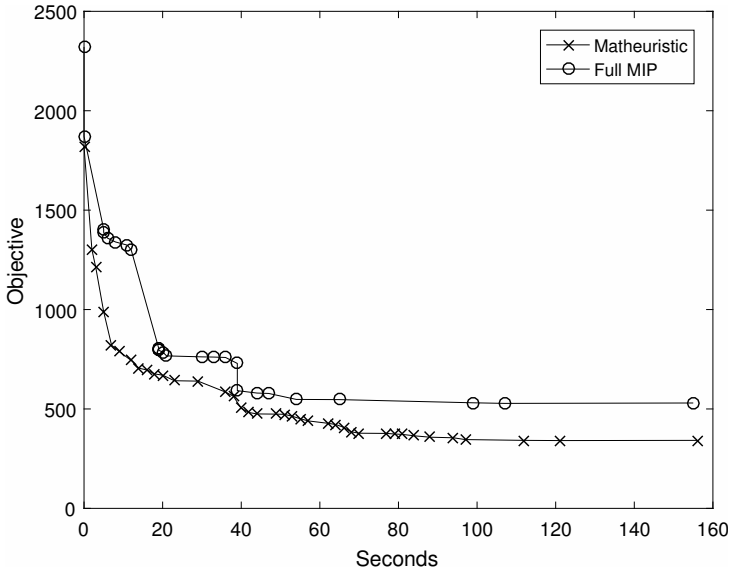


Figure 6.2: Solving Comp12. It can be seen that the matheuristic makes many small improvements compared to the full MIP that hits some large improvement less frequently.

6.5.1 Comparison of the neighborhoods and the full MIP

To see the effects of the three neighborhoods, a comparison is made between the individual performance of each neighborhood, the full matheuristic and solving the full model directly by using a state-of-the-art MIP solver. In Table 6.2 we show the results on solving Stage I of the model. It is seen that each neighborhood by itself does not perform well, especially the assignment neighborhood performs very badly. The *courses* neighborhood works best by itself. It is seen that the performance is improved when two neighborhoods work together, and the algorithm works best when all three neighborhoods are used.

Looking at how the solution improves over time, illustrated in Figure 6.2, it is seen that the profiles differ. The matheuristic makes many small improvements, and the full MIP hits some larger improvements, but these occur less frequently.

Instance	Full MIP	Cou	Cur	Assi	Cou,Cur	Cou,Assi	Cur,Assi	All
comp01	**4.0**	**4.0**	**4.0**	27.3	**4.0**	**4.0**	**4.0**	**4.0**
comp02	50.7	55.4	57.6	159.2	**39.2**	58.9	49.0	45.8
comp03	87.2	98.4	77.8	131.7	**72.1**	92.1	76.9	73.4
comp04	**35.0**	35.3	**35.0**	90.3	**35.0**	**35.0**	**35.0**	**35.0**
comp05	392.2	567.1	**369.6**	687.8	378.1	517.1	379.6	**369.6**
comp06	47.2	57.6	66.9	71.6	43.5	45.6	46.6	**42.4**
comp07	**6.2**	15.1	67.2	23.6	8.0	9.7	10.4	7.8
comp08	**37.0**	37.6	37.0	81.8	**37.0**	38.2	**37.0**	**37.0**
comp09	100.1	100.6	106.9	138.2	99.7	99.8	99.6	**99.1**
comp10	**4.4**	22.1	22.8	59.0	11.4	17.1	11.4	10.2
comp11	**0.0**	**0.0**	**0.0**	**14.2**	**0.0**	**0.0**	**0.0**	**0.0**
comp12	462.5	542.0	366.8	531.5	364.0	418.3	360.4	**359.6**
comp13	**61.4**	68.7	65.5	141.7	61.7	65.2	62.0	61.7
comp14	**51.6**	66.1	61.6	84.8	57.7	59.3	56.5	55.1
comp15	87.2	97.1	76.5	127.6	78.5	92.0	80.3	**73.7**
comp16	**22.1**	36.5	59.9	56.4	29.6	32.8	25.3	26.9
comp17	76.8	79.0	83.5	100.5	**70.7**	75.0	73.4	71.5
comp18	78.9	89.7	76.4	106.7	75.4	76.9	80.7	**71.9**
comp19	**58.1**	70.8	59.1	179.7	59.4	69.0	59.2	59.8
comp20	18.4	41.9	55.9	48.5	16.3	24.1	20.4	**17.7**
comp21	107.2	118.1	107.1	162.1	97.7	105.5	99.5	**95.3**
Avg. Rank	3.0	5.4	4.4	6.7	2.1	4.1	2.9	**1.7**

Table 6.2: Comparison of all combinations of the three neighborhoods Courses, Curriculum, Assignments and the full model on stage I. The best value for each instance is marked with bold. It can be seen that overall the heuristic helps the solver to find better solutions.

6.5.2 Adaptive Neighborhood Sizes

To see the impact of the adaptive neighborhood sizes described in Section 6.4.4, tests are run on the Stage I model with and without the adaptive part. The results are shown in Table 6.3. It is seen that the adaptive neighborhood sizes have a large impact on the results, finding the best solution for 19 of 21 instances. How the adaptive part affects the neighborhood sizes can be seen in Figure 6.3. The two plots show neighborhood sizes and the smoothed MIP gap respectively for the three neighborhoods. It is seen that with the fixed neighborhood sizes, one of the neighborhoods always has a 100% MIP gap, and one always have 0%. With the adaptive part these neighborhood sizes are adjusted so the MIP gap falls between the min. and max. gap of 15% and 20%.

Instance	No Adaptive	Adaptive
comp01	**4.0**	**4.0**
comp02	51.4	**45.8**
comp03	81.9	**73.4**
comp04	**35.0**	**35.0**
comp05	405.6	**369.6**
comp06	43.8	**42.4**
comp07	8.2	**7.8**
comp08	**37.0**	**37.0**
comp09	101.6	**99.1**
comp10	14.5	**10.2**
comp11	**0.0**	**0.0**
comp12	384.9	**359.6**
comp13	**61.2**	61.7
comp14	56.6	**55.1**
comp15	82.6	**73.7**
comp16	28.1	**26.9**
comp17	73.5	**71.5**
comp18	80.5	**71.9**
comp19	**59.3**	59.8
comp20	20.6	**17.7**
comp21	96.5	**95.3**
Avg. Rank	1.7	1.1

Table 6.3: The matheuristic with and without the adaptive part on stage I. It is seen that the adaptive part greatly improves the algorithm and finds the best solution on 19 out of 21 instances.

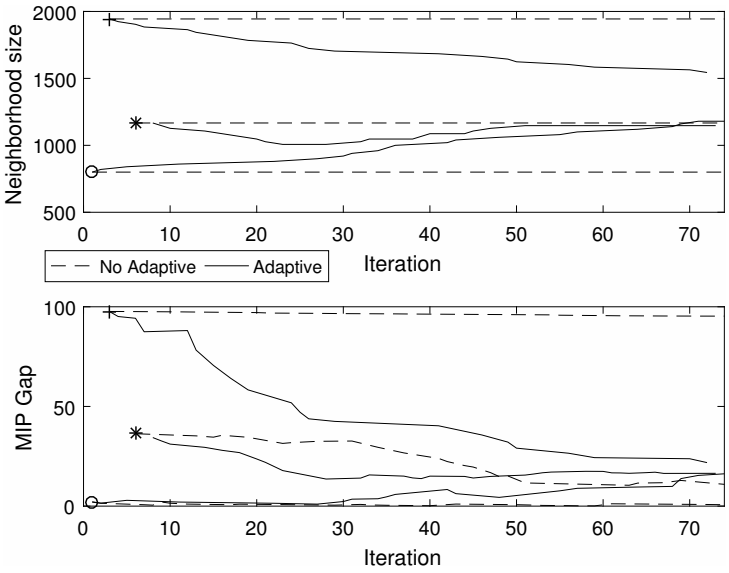


Figure 6.3: The change in neighborhood size and MIP gap for the three neighborhoods with and without the adaptive part. Without the adaptive part two of the MIP Gap's stay at 0% and 100% respectively. The adaptive part changes the size of the neighboorhods so the MIP gaps stay within 15% to 20%.

6.5.3 Comparison with State of the art metaheuristics

To compare with the current state of the art of finding high quality solutions to CB-CTT we compare with Müller [2009], the winner of the ITC2007 contest. We also compare with two more recent algorithms from the literature, which have shown good results, that is a Tabu-based memetic from Abdullah and Turabieh [2012] and a LNS from Kiefer et al. [2014]. The results are shown in Table 6.4.

The matheuristic performs better than the original winner from ITC2007, Müller, but worse than the LNS and the Tabu-based memetic. On one instance the matheuristic finds the best solution.

Instance	Kiefer et al. [2014]	Abdullah and Turabieh [2012]	Müller [2009]	MH
comp01	**5.0**	**5.0**	**5.0**	12.0
comp02	41.9	**36.4**	61.3	49.5
comp03	**72.8**	74.4	94.8	74.5
comp04	**35.2**	38.5	42.8	38.5
comp05	**306.3**	314.5	343.5	373.5
comp06	**48.1**	45.3	56.8	58.3
comp07	15.3	**12.0**	33.9	35.0
comp08	**40.6**	40.8	46.5	49.7
comp09	102.4	108.4	113.1	**100.5**
comp10	13.3	**8.4**	21.3	25.7
comp11	**0.0**	**0.0**	**0.0**	6.5
comp12	323.9	**320.3**	351.6	360.7
comp13	**63.8**	64.3	73.9	69.0
comp14	**56.1**	64.4	61.8	56.9
comp15	73.8	**72.7**	94.8	74.5
comp16	34.8	**23.7**	41.2	37.1
comp17	**73.0**	76.4	86.6	86.1
comp18	**66.5**	75.6	91.7	72.9
comp19	**64.6**	66.8	68.8	64.8
comp20	24.0	**13.5**	34.3	34.3
comp21	**95.3**	100.7	108.0	103.8
Avg. rank	1.4	1.8	3.3	3.1

Table 6.4: Comparison with the matheuristic including stage II with state of the art metaheuristics. The running time is 1 CPU Time unit. The bold results are the best result for that instance. The matheuristic (MH) performs better than the initial winner of the competition but not better than more recent versions.

Very large instances

To see how the algorithm performs on very large instances we use the ones from University of Erlangen. These instances have around six times as many decision variables at the first stage and are far more constrained. Because these instances are much bigger, the timelimit is set to 10 CPU time units. The results can be seen in Table 6.5. For two of the instances the solver applied to the full MIP is not able to find a feasible solution, and in all cases the matheuristic is able to find better solutions. Wbest show the currently best known solution. An example of a run is seen in Figure 6.4 which shows that the solver has difficulties in improving the solution for the full MIP. This shows that the matheuristic is more stable in finding feasible solutions and improving them afterwards.

6.6 Conclusion

We have proposed a matheuristic that works as a framework on top of a MIP model. The approach makes it possible to find good solutions even on larger and more constrained

Instance	Wbest	Full MIP	Matheuristic
erlangen2011_2	4670	-	**5,956.0**
erlangen2012_1	5716	19,067.0	**9,648.8**
erlangen2012_2	8813	-	**15,059.8**
erlangen2013_1	5476	20,467.0	**10,052.0**
erlangen2013_2	8150	16,308.0	**11,120.4**
erlangen2014_1	5981	15,765.0	**8,372.0**
Avg. Rank		2	1

Table 6.5: Comparison of the matheuristic on 6 very large instances running with 10 CPU time units. The results are for the full model (StageI+II) and Wbest is the currently best known solution. The matheuristic outperforms the full mip consistently and is able to find solutions where the solver can not on two of the instances.

instances.

The algorithm has been applied to the standard benchmark datasets for Curriculum-based University Course Timetabling. To the best of our knowledge, this is the first paper which describes a matheuristic for this problem.

Computational results have shown that the matheuristic is better at finding feasible solutions than solving the original MIP with a MIP solver.

Compared to metaheuristics, which is the most used method for practical timetabling, the algorithm performs well. On average, it performs better than the original winner from ITC2007, but not better than two more recently proposed heuristics. Research in heuristics for timetabling has a long history, and more research is needed before matheuristics can catch up. The proposed method can, however, utilize future improvements of both solvers and models, and especially the solvers have improved tremendously over the last decade Bixby [2012]. Furthermore, the matheuristic algorithm has the advantage of being flexible when it comes to adding new constraints and objectives. The authors therefore believe that matheuristics and mathematical programming will become a major approach for practical timetabling.

6.6.1 Outlook

There are a lot of opportunities for future research in the area of matheuristics, aso applied to CB-CTT. An important aspect is to make sure that the computational resources are used to solve the right subproblems. In the following we propose three areas as subjects for future research.

Neighbourhoods It would be interesting to look more into what the characteristics of a good neighborhood are. This would also make it easier to adapt the algorithm to new problems. Another interesting thing would be to generalize it to generic MIPs. This would require neighborhoods with more generic definitions, as opposed to using problem-specific knowledge.

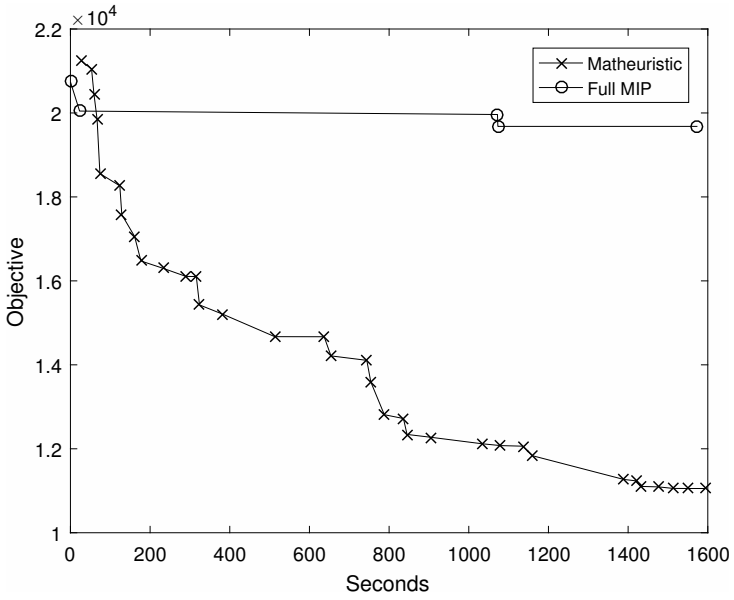


Figure 6.4: Solving the very large instance Erlangen2012_1. The matheuristic finds a worse start solution but is able to improve it compared to the solver on the full model that can not find any improving solution.

More Adaptive parameters The time limit in each iteration and the min. and max. MIP gap is manually set. There might be a potential in making them adaptive. So that for example in the beginning there is a short time limit and high MIP gap, and in the later iterations, where the solution is closer to optimality, the MIP gap in each iteration is smaller, and the time limit is longer.

Parallelization Because of the fixed variables many of the neighborhoods are not connected and can therefore be solved in parallel. This can lead to further speed up by exploiting multiple processors. This would, however, require the processors to somehow share an instance of the MIP model, a problem to which there is no obvious solution in our opinion.

Acknowledgement

The authors would like to thank the organizers of ITC-2007 for providing a formal problem description of CB-CTT as well as benchmark instances. The authors would also like to thank Alex Bonutti, Luca Di Gaspero and Andrea Schaerf for creating and maintaining the website for instances and solutions to CB-CTT.

Bibliography

Salwani Abdullah and Hamza Turabieh. On the use of multi neighbourhood structures within a tabu-based memetic approach to university timetabling problems. *information sciences*, 191:146–168, 2012.

P. Avella, B. D'Auria, S. Salerno, and I. Vasilev. A computational study of local search algorithms for italian high-school timetabling. *Journal of Heuristics*, 13:543–556, 2007. ISSN 1381-1231.

Andrea Bettinelli, Valentina Cacchiani, Roberto Roberti, and Paolo Toth. An overview of curriculum-based course timetabling. *TOP*, pages 1–37, 2015.

Robert E. Bixby. *Optimization Stories*, volume Extra of *21st International Symposium on Mathematical Programming Berlin*, chapter A Brief History of Linear and Mixed-Integer Programming Computation, pages 107–121. Journal der Deutschen Mathematiker-Vereinigung, August 2012.

EK Burke, J Marecek, AJ Parkes, and H Rudová. Uses and abuses of mip in course timetabling. In *Poster at the Workshop on Mixed Integer Programming, MIP2007, Montréal*, 2008.

V. Cacchiani, A. Caprara, R. Roberti, and P. Toth. A new lower bound for curriculum-based course timetabling. *Computers & Operations Research*, 40(10):2466 – 2477, 2013. ISSN 0305-0548.

Marco Caserta and Stefan Voss. Metaheuristics: Intelligent problem solving. In Vittorio Maniezzo, Thomas Statzle, and Stefan Voss, editors, *Matheuristics*, volume 10 of *Annals of Information Systems*, pages 1–38. Springer US, 2010. ISBN 978-1-4419-1305-0.

Emilie Danna, Edward Rothberg, and Claude Le Pape. Exploring relaxation induced neighborhoods to improve mip solutions. *Mathematical Programming*, 102(1):71–90, 2005.

Luca Di Gaspero, Barry McCollum, and Andrea Schaerf. The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3). Technical report, School of Electronics, Electrical Engineering and Computer Science, Queenes University SARC Building, Belfast, United Kingdom, 2007.

Matteo Fischetti and Andrea Lodi. Local branching. *Mathematical Programming*, 98: 23–47, 2003. ISSN 0025-5610.

Jin-Kao Hao and Una Benlic. Lower bounds for the ITC-2007 curriculum-based course timetabling problem. *European Journal of Operational Research*, 212(3):464 – 472, 2011. ISSN 0377-2217.

A. Kiefer, R. Hartl, and A. Schnell. Adaptive large neighborhood search for the curriculum-based course timetabling problem. Technical report, University of Vienna, 2014.

G. Lach and M. Lübbecke. Curriculum based course timetabling: new solutions to udine benchmark instances. *Annals of Operations Research*, 194:255–272, 2012. ISSN 0254-5330.

T. Müller. Itc2007 solver description: a hybrid approach. *Annals of Operations Research*, 172:429–446, 2009. ISSN 0254-5330.

P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In Michael Maher and Jean-Francois Puget, editors, *Principles and Practice of Constraint Programming CP98*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer Berlin / Heidelberg, 1998.

Moshe Sniedovich and Stefan Voß. The corridor method: a dynamic programming inspired metaheuristic. *Control and Cybernetics*, 35(3):551, 2006.

Juan Pablo Vielma. Mixed integer linear programming formulation techniques. *SIAM Review*, 57(1):3–57, 2015.

Chapter 7

Quality Recovering of University Timetabling

Michael Lindahl · Thomas R. Stidsen · Matias Sørensen

Abstract At universities, the timetable plays a large role in the daily life of students and staff, showing when and where lectures are given. But whenever a schedule is executed in a dynamic environment, disruptions will occur. It is then desirable to find a new timetable similar to the old one, so only a few people will be affected. This leads to a minimum perturbation problem, where the goal is to find a feasible timetable by changing as few assignments as possible. This solution will, however, often lead to timetables of low quality.

The purpose of this paper is to show that minimum perturbation solutions have low quality and how using additional perturbations results in timetables with significantly higher quality while still keeping the number of perturbations low.

We formulate a bi-objective model and propose a method to solve it by using mixed integer programming. We test the method on 21 instances of the Curriculum-based Course Timetabling Problem with four different types of disruptions. The us of bi-objective optimization enables us to generate multiple solutions whereby we give the planner a choice. This allows the decision makers to determine the best trade-off between the number of perturbations and the quality, ultimately leading to better timetables for students and staff when disruptions occur.

Keywords University Timetabling · Disruptions · Minimum Pertubation · Mixed integer programming

7.1 Introduction

Disruptions are unavoidable to all schedules executed in a dynamic environment. This is also the case at universities where the timetable determines when and where lectures are taught. After a timetable is finalized and published to lecturers and students, changes will inevitably occur. For example, a lecturer might become unable to teach at a certain time during the week, or a room might get reserved for a conference for an entire day.

When disruptions occur to a published timetable, changing the timetable will cause inconvenience for the effected parties. Therefore, it is desirable that the new timetable is as similar to the old one as possible, so the changes do not affect too many people. This leads to the minimum perturbation problems where the goal is to reconstruct feasibility for an infeasible schedule by making as few perturbations (changes) as possible.

However, creating a feasible timetable is only a part of the goal of the minimum perturbation problem. The quality of the timetable is of great importance to universities. The quality is defined by soft constraints, which are undesired features of the timetable that should be avoided. Variations of these features that occur in university timetabling are described in Bonutti et al. [2012]. Using the solution with the minimum number of perturbations might lead to a big increase in the violation of the soft constraints given. This is undesirable, and it is likely that the planner wants to make additional perturbations to obtain a timetable of higher quality.

The purpose of this paper is to analyze the trade-off between perturbations and quality. This will give us insight into how much minimum perturbation solutions affect the quality of timetables, and how the addition of more perturbations will improve the quality. Providing planners with these trade-offs can assist them in recovering an infeasible timetable and still obtain high-quality timetables.

The paper is organized as follows: In Section 7.2 we describe the perturbation problem and how we model it. In Section 7.3 we will show an algorithm to solve the problem. In Section 7.4 we show the computational results, and finally, in Section 7.5 we will give our conclusions.

7.1.1 Previous work

Minimum perturbation problems in university timetabling have received little attention in the literature compared to the full static problem of creating the entire timetable. The first work on minimum perturbations in scheduling was in El Sakkout and Wallace [2000] who presented an algorithm based on constraint programming to minimally reconfigure a timetable that has become invalid. Later, Elkhyari et al. [2003] also use constraint programming to solve timetabling problems modeled as a minimum perturbation problem on Resource-Constrained Project Scheduling Problems. In Zivan et al. [2011] they proposed a hybrid search, also using constraint programming to solve minimum perturbation problems for scheduling. An improved algorithm, utilizing lower bounds, was proposed in Fukunaga [2013].

The first use of the minimum perturbation apprioach on real university timetabling instances was in Barták et al. [2003], where they propose a branch-and-bound-like algorithm to find approximate solutions. This work is extended in Muller et al. [2005], where they modify the iterative forward search for the static problem to solve minimum

perturbation problems. Finally, in Rudová et al. [2011] they describe their whole system for timetabling at a large university that both incorporates minimal perturbation problems and also problems they call *interactive problems*, where the software suggests small changes to existing timetables.

More recently, Phillips et al. [2016] use mixed integer programming to solve the minimum perturbation problem on real life instances from the University of Auckland. To avoid solving large models they create a smaller problem with only a part of the timetable, and if no feasible solution is found, they gradually expand it until a valid perturbation is found. Mixed-integer programming has however been used on the static university timetabling, see for example Ferland and Roy [1985], Burke et al. [2008], Lach and Lübbecke [2008], Phillips et al. [2015].

For a more broad perspective on dynamic problems we refer to Kocjan [2002] and Verfaillie and Jussien [2005].

Overall, there is a lack of research that examines the loss of quality when choosing the minimal perturbation solution, and what the potential benefit is when using additional perturbations. Mixed Integer Programming (MIP) has only been used in one previous paper to solve this problem, but we believe that MIP models and their solvers will be well suited.

7.2 The quality recovering problem

A perturbation problem consists of three components which are described in this section. In loose terms, the perturbation problem can be defined as follows,

$$\textit{Pertubation problem} = \textit{static problem} + \textit{solution} + \textit{disruption} + \Delta$$

The first part is the static problem, which is the underlying problem formulation needed to create the timetable in the first place (containing all the hard and soft constraints). The second part is the solution i.e., a feasible assignment of the courses with respect to the static formulation. The third part is a disruption that changes the formulation and makes the solution infeasible. Finally, to calculate the similarity between two solutions a perturbation function is used, denoted Δ.

In the quality recovering problem we will distinguish between four solutions defined as follows:

- S_{Init}: The initial solution that after the disruption becomes infeasible.
- S_{QR}^{Δ}: The best solution with the distance Δ from the initial solution. Two special cases of this solution are:
  - $S_{\text{QR}}^{\min}$: The minimum perturbation solution ie. the closest feasible solution, where min is the number of perturbations from the initial solution.
  - $S_{\text{QR}}^{\max}$: The solution with the best quality, furthest away from the initial solution, where max is the number of perturbations from the initial solution.

Figure 7.1 illustrates the relation between these solutions, the solution space and the disruption. In the following we describe the specific parts of the perturbation problem for the university course timetabling problem.

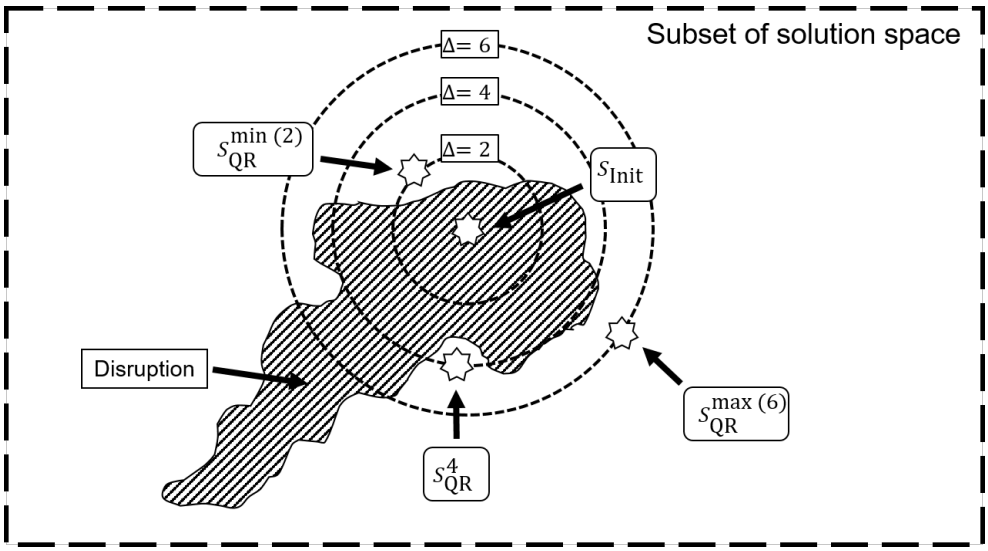


Figure 7.1: An example of the perturbation problem. The figure shows a subset of the solution space. The disruption makes a subspace of the solution space infeasible (grey area), including the initial solution. The circles show two distances from the initial solution. The minimum perturbation solution is the nearest feasible solution to the initial solution. The quality recovering solutions is the one that is further away but has higher quality.

7.2.1 Static problem

The static problem comes from the Second International Timetabling Competition ITC-2007 stated in Di Gaspero et al. [2007], namely Curriculum-based Course Timetabling. The problem was created to allow researchers to compare algorithms and results on the same instances. The problem consists of assigning lectures to timeslots and rooms while taking both hard and soft constraints into account.

All instances are real-world data sets from the University of Udine. The hard and soft constraints originate from this university, but were generalized with the goal of not making the model too complex while still having different types of constraints to capture the essence of real world timetabling. This problem has received much attention in the literature as the de-facto benchmarking data set for university timetabling. For an overview of the literature on this, we refer to the survey by Bettinelli et al. [2015].

$$
\begin{aligned}
\min \quad & f_{qual} = \sum_{c\in\mathcal{C},p\in\mathcal{P},r\in\mathcal{R}} q_{c,r}\cdot x_{c,p,r} && && && \text{(3a)}\\
& + \sum_{c\in\mathcal{C},r\in\mathcal{R}} 1\cdot y_{c,r} - |\mathcal{C}| && && && \text{(3b)}\\
& + \sum_{c\in\mathcal{C}} 5\cdot w_c && && && \text{(3c)}\\
& + \sum_{cu\in\mathcal{CU},p\in\mathcal{P}} 2\cdot v_{cu,p} && && && \text{(3d)}\\
\text{s. t.} \quad & \sum_{p\in\mathcal{P},r\in\mathcal{R}} x_{c,p,r} && = L(c) && \forall c\in\mathcal{C} && \text{(3e)}\\
& \sum_{c\in\mathcal{C}} x_{c,p,r} && \leq 1 && \forall p\in\mathcal{P}, r\in\mathcal{R} && \text{(3f)}\\
& \sum_{c\in\mathcal{C}(t),r\in\mathcal{R}} x_{c,p,r} && \leq 1 && \forall t\in\mathcal{T}, p\in\mathcal{P} && \text{(3g)}\\
& \sum_{c\in\mathcal{C}(cu),r\in\mathcal{R}} x_{c,p,r} && \leq 1 && \forall cu\in\mathcal{CU}, p\in\mathcal{P} && \text{(3h)}\\
& \sum_{p\in\mathcal{P}} x_{c,p,r} - |\mathcal{P}|\cdot y_{c,r} && \leq 0 && \forall c\in\mathcal{C}, r\in\mathcal{R} && \text{(3i)}\\
& \sum_{p\in d,r\in\mathcal{R}} x_{c,p,r} - z_{c,d} && \geq 0 && \forall c\in\mathcal{C}, d\in\mathcal{D} && \text{(3j)}\\
& \sum_{d\in\mathcal{D}} z_{c,d} + w_c && \geq mnd(c) && \forall c\in\mathcal{C} && \text{(3k)}\\
& \sum_{c\in\mathcal{C}(cu),r\in\mathcal{R}} x_{c,p,r} - r_{cu,p} && = 0 && \forall cu\in\mathcal{CU}, p\in\mathcal{P} && \text{(3l)}\\
& - r_{cu,p-1} + r_{cu,p} - r_{cu,p+1} - v_{cu,p} && \leq 0 && \forall cu\in\mathcal{CU}, p\in\mathcal{P} && \text{(3m)}\\
& x_{c,p,r}\in\mathbb{B} && && \forall c\in\mathcal{C}, p\in\mathcal{P}, r\in\mathcal{R} && \text{(3n)}\\
& y_{c,r}\in\mathbb{B} && && \forall c\in\mathcal{C}, r\in\mathcal{R} && \text{(3o)}\\
& w_c\in\mathbb{Z}_+ && && \forall c\in\mathcal{C} && \text{(3p)}\\
& z_{c,d}\in\mathbb{B} && && \forall c\in\mathcal{C}, d\in\mathcal{D} && \text{(3q)}\\
& v_{cu,p}\in\mathbb{B} && && \forall cu\in\mathcal{CU}, p\in\mathcal{P} && \text{(3r)}\\
& r_{cu,p}\in\mathbb{B} && && \forall cu\in\mathcal{CU}, p\in\mathcal{P} && \text{(3s)}
\end{aligned}
$$

Model 3: The MIP model for the static university timetabling problem.

First, we will describe the problem and explain the model by using mixed-integer programming. In Burke et al. [2008] they proposed the first model known as the monolithic model or the three-index model, seen in Model 3. First, the hard constraints given in (3e)-(3h) are described and then the soft constraints given in (3i)-(3m).

Hard constraints

In curriculum-based course timetabling there is given a list of courses, $\mathcal{C}$, a list of timeslots, $\mathcal{P}$, and a list of rooms, $\mathcal{R}$. A course $c \in \mathcal{C}$ consists of a number of lectures, $L(c)$. The goal is then to schedule all lectures by assigning them to a timeslot and a room. For this we use the following binary decision variable that determines which assignments are used

$$x_{c,p,r} = \begin{cases} 1 & \text{if course } c \in \mathcal{C} \text{ is planned at period } p \in \mathcal{P} \text{ and in room } r \in \mathcal{R} \\ 0 & \text{otherwise} \end{cases}$$

First of all, every lecture needs to be scheduled, which is handled by constraint (3e). A room can only fit one course in it, which is ensured by constraint (3f). We also have a set of teachers $\mathcal{T}$, where $\mathcal{C}(t)$ is the set of courses taught by teacher $t \in \mathcal{T}$. A teacher can only teach one course at the time ensured by constraint (3g). Furthermore, students can also only be at one place at a time. Therefore, a set of curricula $\mathcal{CU}$ is given and courses that are a part of the same curriculum, $\mathcal{C}(cu)$, cannot be placed in the same timeslot, ensured by constraint (3h). Last, there are unavailabilities, which are timeslots to which specific courses cannot be assigned. We will set $x_{c,p,r} = 0$ if course $c \in \mathcal{C}$ cannot be taught in timeslot $p \in \mathcal{P}$.

Soft constraints

The quality of a timetable is measured by how many soft constraints are being violated. In total four soft constraints are defined, and each one is associated with a number of penalty points. The objective function (3a)-(3d) is then equal to the sum of these penalty points. The soft constraints are as following:

RoomCapacity Each room is associated with a capacity, and each course is associated with a number of students attending the course. A penalty of *1* is given for each extra student assigned to a room. This is calculated in the objective (3a), where we set $q_{c,r}$ equal to the undercapacity of room r if course c is assigned to it.

RoomStability All lectures from the same course are assigned to the same room. A penalty of *1* is given for each extra room used. To calculate this we introduce the binary variable $y_{c,r}$ to indicate if course c is scheduled in room r, as ensured by constraint (3i). The violation is then calculated in objective (3b) as the sum of rooms used minus the total number of courses.

MinimumWorkingDays The workload of a course should be distributed throughout the week. Each course, c, therefore, has a number of minimum working days, $mnd(c)$, on which it at least should be planned. The binary variable $z_{c,d}$ indicates if course c is scheduled on day d, ensured by constraint (3j). The amount of violation, w_c, for course c is then calculated by constraint (3k) and summed up in objective (3c) with the penalty of *5*.

CurriculumCompactness Students should not have idle timeslots, and therefore it is desirable that a lecture from a given curriculum is next to a lecture from the same curriculum. *Two* penalty points are given for each violation. The binary variable $r_{cu,p}$ indicates if a lecture from curriculum cu is planned in timeslot p, ensured by constraint (3l). The variable $v_{cu,p}$ then indicates if curriculum cu has no neighbors in timeslot p, ensured by constraint (3m) and added in the objective in (3d).

7.2.2 Disruption

In this section we introduce the concept of disruption in terms of the model described in Section 7.2.1. Disruptions take different forms depending on the type of environment in which they occur, but are always related to a resource, e.g. a room, teacher, student or a timeslot.

In this paper we will consider two types of disruptions: 1) A resource is removed i.e. unavailable for assignment, and 2) a new shared resource is added, i.e. some courses share a resource and cannot be assigned to the same timeslot. These two very generic disruptions types show that MIP can represent the different kinds of disruptions that occur at a university. A removed resource is for example when a room is unavailable or a course cannot be taught in a specific timeslot. This is modeled in the following way by altering the MIP model. Let J be the set of tuples with all combinations of courses, timeslots and rooms.

$$J = \{(c, p, r) : c \in \mathcal{C},\, p \in \mathcal{P},\, r \in \mathcal{R}\} \tag{7.2}$$

Then let $\hat{J} \subset J$ be the set of resource combinations that are removed and fix those variables to zero in the model, thereby making those solutions infeasible:

$$x_{c,p,r} = 0 \quad \forall (c, p, r) \in \hat{J} \tag{7.3}$$

Similar to removing a resource, a new shared one could also be added. This is for example the case if a new curriculum is added because a group of students needs to be able to take a specific set of courses. This will result in a conflict if these courses are taught at the same time. Let $\hat{\mathcal{C}} \subset \mathcal{C}$ be the courses that share the new resource. The following constraint is then added to the model,

$$\sum_{c \in \hat{\mathcal{C}}, r \in \mathcal{R}} x_{c,p,r} \leq 1 \quad \forall p \in \mathcal{P} \tag{7.4}$$

Both of the two disruption types restrict the static model. This means that a new solution, S^{Δ}_{QR}, can never be better than the initial solution, S_{Init}, as a feasible solution to the dynamic problem also is a feasible solution to the static problem, and the objective is the same.

$$
\begin{aligned}
\min \quad & f_{qual} && \text{(as defined by (3a)-(3d))} && \text{(8a)}\\
& f_{\Delta} = \Delta(x, \bar{x}) && \text{(as defined by (7.5))} && \text{(8b)}\\
\text{s. t.} \quad & (3e) - (3s) && && \text{(8c)}\\
& \text{disruption} && \text{(any of (7.3)-(7.4))} && \text{(8d)}
\end{aligned}
$$

Model 8: The bi-objective MIP model for the quality recovering problem, consisting of the static model with an additional objective and a disruption.

7.2.3 Perturbation function

To measure the amount of changes between the initial solution, S_{Init}, and a new solution, S^{Δ}_{QR}, we need a perturbation function that will be our second objective. For this we use the hamming distance to calculate the amount of decision variables of which the value have changed. Because the sum of all variables is always the same, due to constraint (3e), we only calculate the values that change from one to zero. Let $\bar{J} = \{(c, p, r) \in J : \bar{x}_{c,p,r} = 1\}$. The hamming distance between two solutions are defined as,

$$\Delta(x, \bar{x}) = \sum_{(c,p,r)\in\bar{J}} (1 - x_{c,p,r}) \tag{7.5}$$

7.2.4 Quality recovering model

Altering the original static model by adding a disruption and a new minimum perturbation objective results in the bi-objective model shown in Model 8. $\bar{x}$ is the solution from S_{init}. All constraints and variables from the static problem is added in constraint (8c) and in the objective (8a). The perturbation measure is added as a second objective (8b). Finally, we add the disruption in constraint (8d). As mentioned, this disruption can take the different forms described in Section 7.2.2. The two solutions $S^{\min}_{\text{QR}}$, $S^{\max}_{\text{QR}}$ are equal to the two lexicographic solutions, i.e. minimizing the objectives in prioritized order.

7.3 Quality recovering algorithm

To solve the quality recovering problem in Model 8 we need to take both objectives into account; the quality objective f_{qual} and the perturbation objective f_{Δ}. These two objectives are conflicting, as keeping the number of perturbations down limits the potentially improving solutions that can be found. This results in a bi-objective optimization problem. To solve this we search for pareto-optimal solutions, which are defined as solutions where one objective cannot be improved without worsening the other.

To generate this solution frontier, we base our algorithm on the ϵ-constraint method from Haimes et al. [1971]. The approach is to put a constraint on one of the objectives while minimizing the other and repeat this with a new constraint. The pseudo code is seen in Algorithm 7.3.1. The method starts by solving the minimum perturbation problem

and then put a constraint on the perturbation objective. It then iteratively increases the allowed number of perturbations while minimizing the quality objective.

The algorithm is terminated when no improved solution can be found, meaning that a single extra perturbation does not lead to an improvement in f_{qual}. Therefore, it is possible that an improvement in quality occurs if additional extra perturbations are used. Finding the minimum quality without the perturbation constraint is the same as solving the static problem to optimality, which is very difficult (see Cacchiani et al. [2013]).

In practice, additional stopping criteria can be used to keep the running time down. This could be a time limit or a upper bound on the number of iterations.

The algorithm is simple and fully sufficient for our problem. Because it uses mixed integer programming underneath, it is very generic and can be used with most kind of static problems as well as different perturbation functions depending on the problem at hand.

Algorithm 7.3.1 Quality Recovering

1: $\tilde{\Delta} \leftarrow Minimize(f_\Delta)$ ▷ Minimum Pertubations
2: $\tilde{f}_{qual} \leftarrow \infty$
3: **repeat**
4: Update ϵ-constraint: $\tilde{f}_\Delta = \tilde{\Delta}$
5: $S^{\tilde{\Delta}}_{\text{QR}} \leftarrow Minimize(f_{qual})$ ▷ Find Pareto-solution
6: $\tilde{\Delta} \leftarrow \tilde{\Delta} + 1$
7: **until** No improving solution is found

7.4 Computational results

To show the applicability of our method we use the data sets from the Second International Timetabling Competition, described in Di Gaspero et al. [2007]. The 21 data sets can be seen in Table 7.1, and it is seen that there is a variance between sizes and curricula. We will use the currently best-known solution for each instance for the initial solution, S_{Init}, and then disrupt it so it becomes infeasible. All data sets and best-known solutions can be found at `http://tabu.diegm.uniud.it/ctt/`. The complete source-code to make the computations are available on `http://github.com/miclindahl/UniTimetabling`.

We will analyze the impact of four different disruptions that are very different, both in terms of the way they impact the timetable and in terms of the number of lectures affected. These disruptions cover scenarios that usually happen at a university. The four disruptions are the following:

One Assignment Invalid One assignment, $(\hat{c}, \hat{p}, \hat{r})$, is invalid and needs to be assigned to a different room or to a different timeslot. The following constraint is added

$$x_{\hat{c},\hat{p},\hat{r}} = 0$$

Insert Curriculum Takes four courses $\hat{\mathcal{C}}$ and put them into a curriculum meaning that they cannot be taught at the same time. To be able to compare with S_{Init} the

Instance	Courses	Rooms	Timeslots	Curricula
comp01	30	6	30	14
comp02	82	16	25	70
comp03	72	16	25	68
comp04	79	18	25	57
comp05	54	9	36	139
comp06	108	18	25	70
comp07	131	20	25	77
comp08	86	18	25	61
comp09	76	18	25	75
comp10	115	18	25	67
comp11	30	5	45	13
comp12	88	11	36	150
comp13	82	19	25	66
comp14	85	17	25	60
comp15	72	16	25	68
comp16	108	20	25	71
comp17	99	17	25	70
comp18	47	9	36	52
comp19	74	16	25	66
comp20	121	19	25	78
comp21	94	18	25	78

Table 7.1: The 21 instances used from the second international timetabling competition defined in Di Gaspero et al. [2007]. They differ in size and in number of curricula.

objective function is not changed. Only the following constraint is added:

$$\sum_{c \in \hat{\mathcal{C}}, r \in \mathcal{R}} x_{c,p,r} \leq 1 \quad \forall p \in \mathcal{P}$$

Remove Room Whole Day Removes a room $\hat{r}$ for an entire day $\hat{d}$, ie. adding constraint:

$$x_{c,p,\hat{r}} = 0 \quad \forall c \in \mathcal{C}, p \in \hat{d}$$

One Timeslot Unavailable Makes one timeslot $\hat{p}$ unavailable for all courses. The following constraint is added:

$$x_{c,\hat{p},r} = 0 \quad \forall c \in \mathcal{C}, r \in \mathcal{R}$$

All computations are made on a 64 bit Windows machine with a 4 GHz Intel Core i7 CPU and 32 GB of memory. To solve the integer programs we use Gurobi 7.0 with standard settings. All problems are solved to optimality, but an upper limit of 20 perturbations is set to keep the computational time reasonable. We will first analyze each disruption independently and show the impact on all instances, and, finally, we will summarize all the results and look into the computational difficulty.

7.4.1 One assignment invalid

The resulting full pareto fronts from making one assignment invalid on each of the 21 datasets is seen in Figure 7.2. Table 7.2 summarizes the results and shows the two lexicographic solutions, S_{QR}^{min} and S_{QR}^{max}. There is a significant difference between the data sets where the three instances comp16, comp17, and comp21 only needs one perturbation to obtain a feasible solution with the same quality as the disrupted solution. The data sets comp09 and comp15 are different. Even though they can also be made feasible using only one perturbation, they require four extra perturbations to recover the quality, which is still not the same level of quality as before the disruption.

Instance	S_{Init}	$S_{\mathrm{QR}}^{\mathrm{min}}$		$S_{\mathrm{QR}}^{\mathrm{max}}$	
	f_{qual}	f_Δ	f_{qual}	f_Δ	f_{qual}
comp01	5	1	41	2	6
comp02	24	1	33	2	32
comp03	64	1	70	2	†64
comp04	35	2	36	2	36
comp05	285	1	292	2	289
comp06	27	1	32	3	28
comp07	6	1	11	3	8
comp08	37	1	41	1	41
comp09	96	1	101	5	97
comp10	4	1	10	2	7
comp11	0	1	4	2	1
comp12	294	1	312	3	†294
comp13	59	1	183	3	†59
comp14	51	1	53	2	†51
comp15	62	1	89	5	64
comp16	18	1	†18	1	†18
comp17	56	1	†56	1	†56
comp18	61	1	70	4	65
comp19	57	1	59	2	†57
comp20	4	2	68	3	6
comp21	74	1	†74	1	†74

Table 7.2: The two lexicographic solutions for the *one assignment invalid* disruption. Eight instances recover completely and regain the same quality level as the disrupted, marked with †.

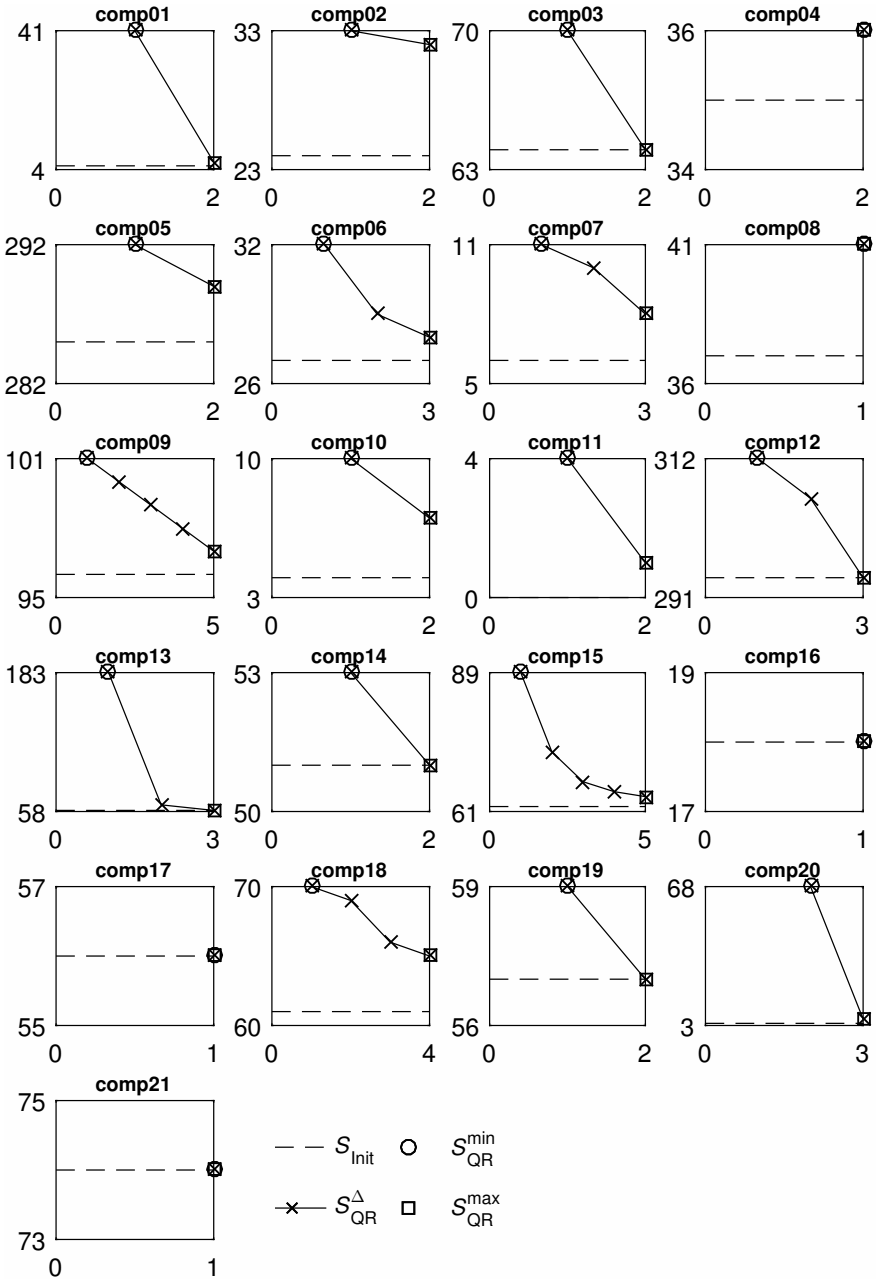


Figure 7.2: The pareto fronts for each data set on the *one assignment invalid* disruption and the objective value of the disrupted solution. The x-axis is f_Δ and the y-axis is f_{qual}. There is a large difference between how much worse the quality is and how much is gained by using extra perturbations.

7.4.2 Insert curriculum

Adding a new curriculum of four courses impacts the timetable on each instance differently. Figure 7.3 shows the pareto fronts together with the value of the disrupted solution. The two instances comp06 and comp19 recovers a lot of quality by allowing extra perturbations. A summary showing the two lexicographic solutions is seen in Table 7.3. Six instances can recover and regain the same value of f_{qual} as before the disruption.

Instance	S_{Init}	$S_{\text{QR}}^{\min}$		$S_{\text{QR}}^{\max}$	
	f_{qual}	f_Δ	f_{qual}	f_Δ	f_{qual}
comp01	5	1	8	3	†5
comp02	24	5	73	7	†24
comp03	64	1	65	1	65
comp04	35	2	39	3	36
comp05	285	1	288	2	287
comp06	27	3	54	9	32
comp07	6	2	10	3	9
comp08	37	5	47	8	39
comp09	96	4	107	7	97
comp10	4	2	22	5	8
comp11	0	3	3	4	†0
comp12	294	1	301	1	301
comp13	59	3	†59	3	†59
comp14	51	1	55	3	†51
comp15	62	3	66	5	†62
comp16	18	2	25	3	24
comp17	56	3	59	5	57
comp18	61	2	82	3	62
comp19	57	4	89	12	63
comp20	4	2	12	3	6
comp21	74	2	88	3	80

Table 7.3: The two lexicographic solutions for each instance with the *insert curriculum* disruption. On six instances the same solution value is completely recovered to the same value as the disrupted solution, marked with †.

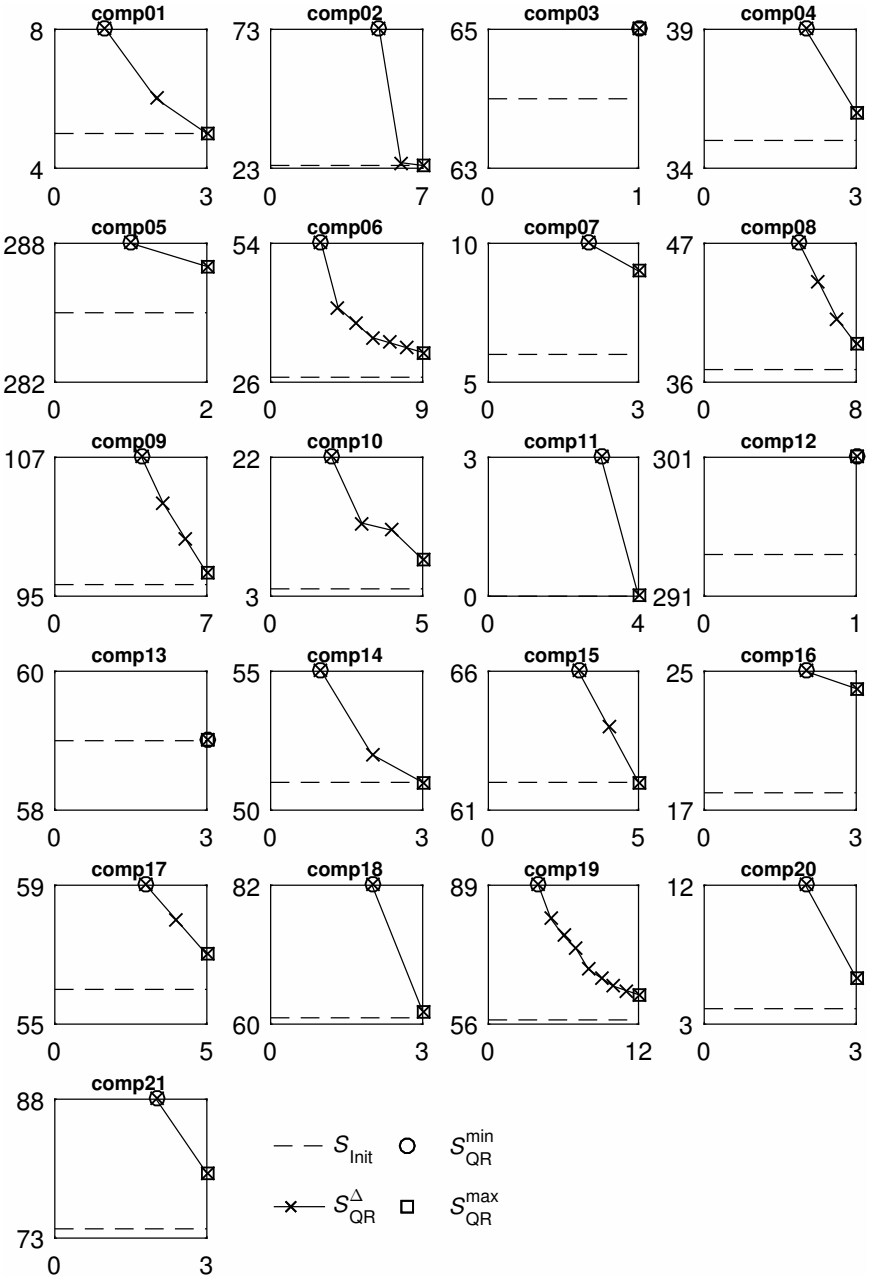


Figure 7.3: The pareto fronts for the *insert curriculum* disruption. The x-axis is f_Δ and the y-axis is f_{qual}. Especially the two instances comp06 and comp19 recover a lot of quality from extra perturbations.

7.4.3 Remove room whole day

The impact of removing a room for a whole day have a great impact on the solution, as shown in Figure 7.4. Especially comp01 loses a lot of quality, going from five to 409 penalty points. The reason for this is that comp01 is the smallest instance with only six rooms, and therefore removing one has a large impact. It is also interesting that on nine of the instances there is no benefit of extra perturbations. Table 7.4 summarizes the results showing that none of the instances recovers the initial objective value.

Instance	S_{Init}	$S_{\text{QR}}^{\min}$		$S_{\text{QR}}^{\max}$	
	f_{qual}	f_Δ	f_{qual}	f_Δ	f_{qual}
comp01	5	6	409	17	92
comp02	24	4	26	5	25
comp03	64	2	65	2	65
comp04	35	2	37	2	37
comp05	285	4	332	8	294
comp06	27	3	38	4	31
comp07	6	3	9	3	9
comp08	37	2	39	2	39
comp09	96	3	99	4	98
comp10	4	5	18	7	8
comp11	0	9	48	14	21
comp12	294	5	299	7	297
comp13	59	2	61	2	61
comp14	51	3	54	5	52
comp15	62	2	64	3	63
comp16	18	5	21	5	21
comp17	56	4	60	4	60
comp18	61	5	98	11	78
comp19	57	3	60	4	59
comp20	4	4	8	4	8
comp21	74	4	78	4	78

Table 7.4: The two lexicographic solutions for the *remove room whole day* disruption. None of the instances recovers to the same value as the disrupted solution.

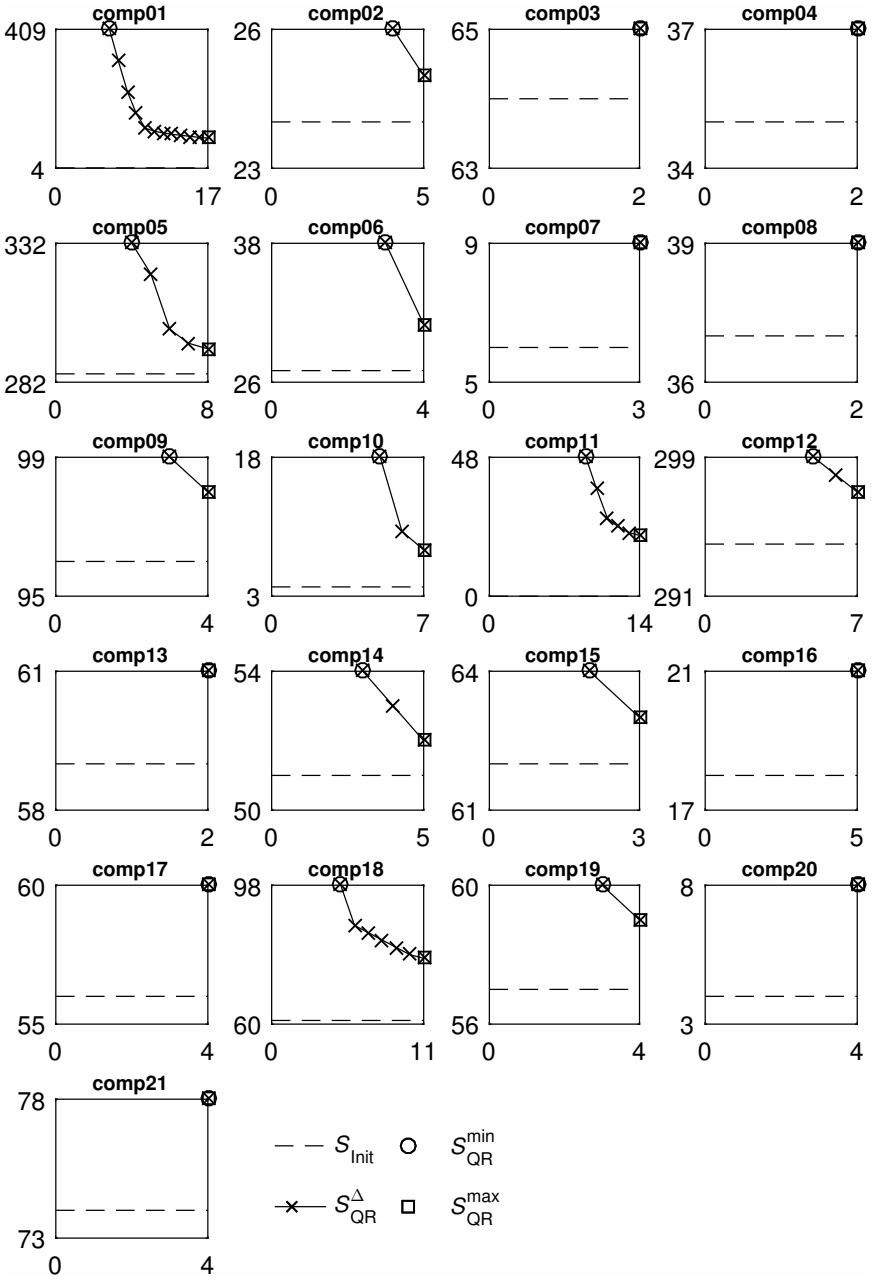


Figure 7.4: The pareto front for the *remove room whole day* disruption. The x-axis is f_Δ and the y-axis is f_{qual}. Especially comp01 is affected and uses many extra perturbations to recover most of the quality.

7.4.4 One timeslot unavailable

The final disruption we investigate is where an entire timeslot is made unavailable. Table 7.5 summarizes the results and show that comp05 is infeasible and cannot be recovered. The other instances require between four and 19 perturbations to become feasible, which shows how large the impact of this disruption is. Figure 7.5 shows the full pareto fronts. Using more perturbations improves the quality a lot on all instances, but most of them are still far away from their initial solution value. Notice that we set an upper limit on perturbations to 20, and therefore it is likely that the quality would be improved by allowing more perturbations, which also would have resulted in longer running times.

Instance	S_{Init}	$S_{\text{QR}}^{\min}$		$S_{\text{QR}}^{\max}$	
	f_{qual}	f_Δ	f_{qual}	f_Δ	f_{qual}
comp01	5	7	44	17	13
comp02	24	16	324	20	171
comp03	64	17	139	20	125
comp04	35	15	100	20	74
comp05	285	-	-	-	-
comp06	27	19	192	20	186
comp07	6	19	194	20	188
comp08	37	14	123	20	96
comp09	96	12	178	20	128
comp10	4	17	118	20	100
comp11	0	4	16	10	†0
comp12	294	10	430	20	358
comp13	59	15	275	20	128
comp14	51	16	148	20	109
comp15	62	14	178	20	132
comp16	18	17	177	20	155
comp17	56	16	149	20	128
comp18	61	8	108	16	96
comp19	57	18	230	20	151
comp20	4	18	334	20	194
comp21	74	18	212	20	198

Table 7.5: The summary of the *one timeslot unavailable* disruption. comp05 is infeasible, and for the other instances, it takes between four and 19 perturbations to find a feasible solution. Notice that we set an upper limit on 20 perturbations, and therefore it is likely that the quality would be improved by increasing this limit.

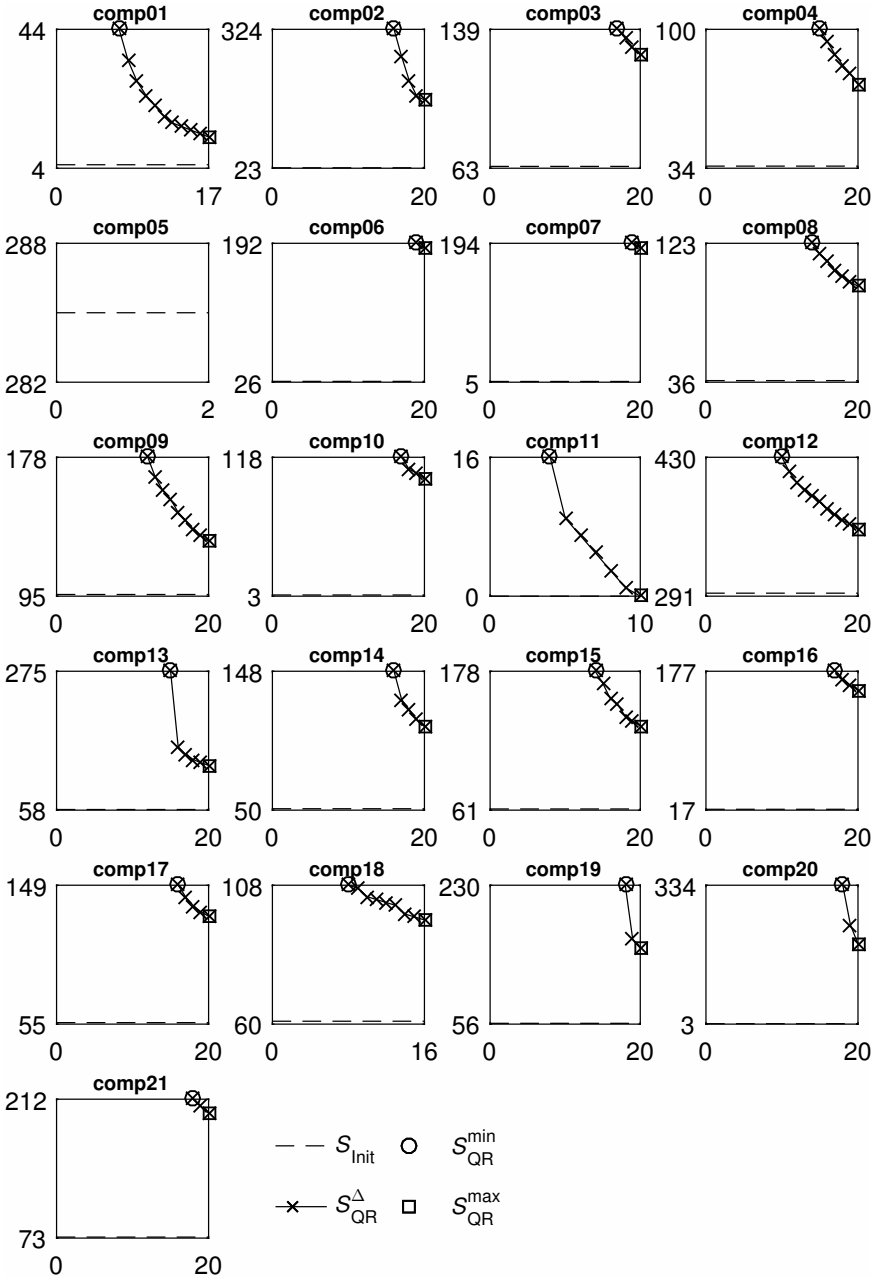


Figure 7.5: The pareto fronts for the *one timeslot unavailable* disruption. The x-axis is f_Δ and the y-axis is f_{qual}. The impact is high on all instances and comp05 is infeasible and cannot be repaired.

7.4.5 Overall comparison

As seen in the previous sections, the impact of the disruptions differs a lot between the disruption types and the data sets. In this section we will summarize the overall results.

Table 7.6 shows the average minimum number of perturbations across all datasets for each of the disruptions. It shows that the amount of perturbations needed to make the solution feasible differs a lot between the disruptions, from 1.1 to 14.5. This means that there is a wide variety of impact of the chosen disruptions on solutions.
Figure 7.6 shows that using additional perturbations decreases the number of violated soft constraints. For each dataset, the results are scaled so the x-axis is the number of extra perturbations relative to the minimum perturbation solution, meaning that zero is equal to the minimum perturbation solution i.e.

$$\bar{f}_{\Delta} = f_{\Delta}(S^{\Delta}_{\mathrm{QR}}) - f_{\Delta}(S^{\min}_{\mathrm{QR}}) \tag{7.7}$$

The y-axis shows the decrease in quality relative to the initial solution, so 0% means that it reaches the same value, and 100% means that it is doubled, i.e.

$$\bar{f}_{qual} = 100 \cdot \left(\frac{f_{qual}(S^{\Delta}_{\mathrm{QR}})}{f_{qual}(S_{\mathrm{Init}})} - 1 \right) \tag{7.8}$$

To summarize on all the 21 instances we use quartiles due to the fact that the variation between the instances is high, and the distribution is skewed with a few very large values. The three lines indicate the 25%, 50% (median) and 75% quartile for the 21 instances.

Figure 7.6 shows that the median quality decrease for the minimum perturbation solution differs between 7% and up to 232% on the disruptions. It is also seen that the 25% most affected instances decreases in quality between 53% and 975%. The loss of quality is, therefore, significant in the minimum perturbation solution.

It is also seen that the quality increases significantly when introducing extra perturbations. For the median it is between 2% and 88%, and on the 75% quartile the improvement is between 19% and 374%. The distribution is skewed as the 25% and 50% quartiles are close, but the 75% is much further away. This shows that a few instances are heavily influences by the disruption compared to others.

Disruption	Avg. Min. Perturbation (f_{Δ})
One Assignment Invalid	1.1
Insert Curriculum	2.5
Remove Room Whole Day	3.8
One Timeslot Unavailable	14.5

Table 7.6: The average number of perturbations to make a solution feasible after each of the four disruptions.

7.4.6 Computational complexity

Solving the static curriculum-based course timetabling problems to optimality is difficult - as seen in Cacchiani et al. [2013] and Bettinelli et al. [2015] - because of the high

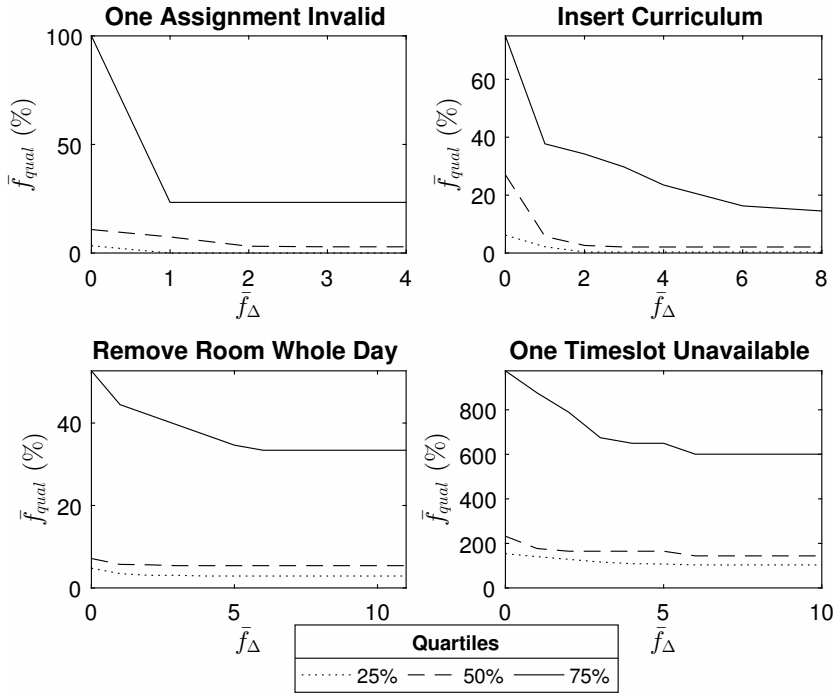


Figure 7.6: An illustration of the quality decrease in the minimum perturbation solution and how it increases by using extra perturbations for each of the different disruptions. The three lines show the quartiles that indicate that there is a significant variance between the datasets.

computational complexity. Adding a constraint on the perturbations reduces the solution space. When the constraint is iteratively relaxed, the computational complexity increases. Figure 7.7 shows the running time for solving the model as a function of the number of perturbations for each of the four different disruptions. Overall is 77% of the iterations solved in less than 10 seconds. But as the number of perturbations increase so does the running time that for many perturbations is as high as 6 hours.

7.5 Conclusion

In this paper, we have proposed a method to recover feasibility of disrupted university timetables while taking quality of solutions into account. We have proposed a bi-objective optimization algorithm to find pareto-optimal solutions. This approach gives the planner the option to choose between several solutions and decide on the best trade-off between finding a timetable similar to the previous one and one with high quality.

Mixed Integer Programming is well suited for this task as it can prove infeasibility and find minimum perturbation solutions fast. However, finding optimal solutions for many additional perturbations is computationally very hard and will often take too much

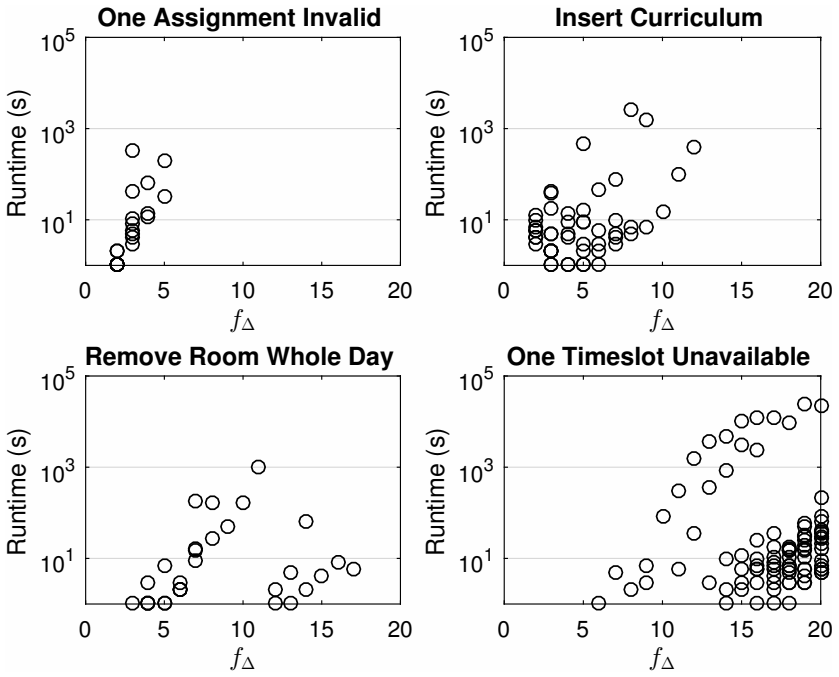


Figure 7.7: Running times for each iteration as a function of the number of perturbations for each of the four different disruptions. In total 77% of the models are solved in less than 10 seconds. But when the number of perturbations increases the solution time can get up to 6 hours.

time to be used in practice. It should be noted that in practice the maximum number of perturbations will often be limited, or a near-optimal solution will be usable, making the proposed approach applicable. This is one of the very first studies that uses mixed integer programming.

We tested the method on 21 instances of the Curriculum-based Course Timetabling problem with four different types of disruptions. This showed how minimum perturbation solutions lead to timetables of low quality with a median decrease in quality between 7% and 232%, depending on the disruption. It was shown how extra perturbations can recover much of the lost quality, between 2% and 88% for the median.

The proposed method is very generic, and the algorithm can be used on a broad variety of timetabling problems after formulating the static problem and the perturbation function using mixed-integer programming.

Acknowledgement

We would like to thank the organizers of the Second International Timetabling Competition for formulating the Curriculum-Based Course Timetabling problem. We would

also like to thank Alex Bonutti, Luca Di Gaspero and Andrea Schaerf for creating and maintaining the website with instances and best-known solutions.

Bibliography

Roman Barták, Tomáš Müller, and Hana Rudová. A new approach to modeling and solving minimal perturbation problems. In *International Workshop on Constraint Solving and Constraint Logic Programming*, pages 233–249. Springer, 2003.

Andrea Bettinelli, Valentina Cacchiani, Roberto Roberti, and Paolo Toth. An overview of curriculum-based course timetabling. *TOP*, pages 1–37, 2015.

A. Bonutti, F. De Cesco, L. Di Gaspero, and A. Schaerf. Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results. *Annals of Operations Research*, 194(1):59–70, April 2012. ISSN 0254-5330.

Edmund K. Burke, Jakub Mareček, Andrew J. Parkes, and Hana Rudová. Penalising patterns in timetables: Novel integer programming formulations. In Jörg Kalcsics and Stefan Nickel, editors, *Operations Research Proceedings 2007*, volume 2007 of *Operations Research Proceedings*, pages 409–414. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-77903-2. 10.1007/978-3-540-77903-2_63.

V. Cacchiani, A. Caprara, R. Roberti, and P. Toth. A new lower bound for curriculum-based course timetabling. *Computers & Operations Research*, 40(10):2466 – 2477, 2013. ISSN 0305-0548.

Luca Di Gaspero, Barry McCollum, and Andrea Schaerf. The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3). Technical report, School of Electronics, Electrical Engineering and Computer Science, Queenes University SARC Building, Belfast, United Kingdom, 2007.

Hani El Sakkout and Mark Wallace. Probe backtrack search for minimal perturbation in dynamic scheduling. *Constraints*, 5(4):359–388, 2000. ISSN 1383-7133. doi: 10.1023/A:1009856210543. URL `http://dx.doi.org/10.1023/A%3A1009856210543`.

Abdallah Elkhyari, Christelle Gueret, and Narendra Jussien. Solving dynamic resource constraint project scheduling problems using new constraint programming tools. In Edmund Burke and Patrick De Causmaecker, editors, *Practice and Theory of Automated Timetabling IV*, volume 2740 of *Lecture Notes in Computer Science*, pages 39–59. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-40699-0.

Jacques A Ferland and Serge Roy. Timetabling problem for university as assignment of activities to resources. *Computers & operations research*, 12(2):207–218, 1985.

Alex Fukunaga. An improved search algorithm for min-perturbation. In *International Conference on Principles and Practice of Constraint Programming*, pages 331–339. Springer, 2013.

Yacov Y Haimes, LS Ladson, and David A Wismer. Bicriterion formulation of problems of integrated system identification and system optimization, 1971.

Waldemar Kocjan. Dynamic scheduling. state of the art report. *SICS Research Report*, 2002.

G. Lach and M. Lübbecke. Optimal university course timetables and the partial transversal polytope. In Catherine McGeoch, editor, *Experimental Algorithms*, volume 5038 of *Lecture Notes in Computer Science*, pages 235–248. Springer Berlin / Heidelberg, 2008.

Tomáš Muller, Hana Rudová, and Roman Barták. Minimal perturbation problem in course timetabling. In Edmund Burke and Michael Trick, editors, *Practice and Theory of Automated Timetabling V*, volume 3616 of *Lecture Notes in Computer Science*, pages 126–146. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-30705-1.

Antony E. Phillips, Hamish Waterer, Matthias Ehrgott, and David M. Ryan. Integer programming methods for large-scale practical classroom assignment problems. *Computers & Operations Research*, 53(0):42 – 53, 2015. ISSN 0305-0548. doi: http://dx.doi.org/10.1016/j.cor.2014.07.012. URL `http://www.sciencedirect.com/science/article/pii/S0305054814001956`.

Antony E Phillips, Cameron G Walker, Matthias Ehrgott, and David M Ryan. Integer programming for minimal perturbation problems in university course timetabling. *Annals of Operations Research*, pages 1–22, 2016.

Hana Rudová, Tomáš Muller, and Keith Murray. Complex university course timetabling. *Journal of Scheduling*, 14(2):187–207, 2011. ISSN 1094-6136.

Gérard Verfaillie and Narendra Jussien. Constraint solving in uncertain and dynamic environments: A survey. *Constraints*, 10(3):253–281, 2005.

Roie Zivan, Alon Grubshtein, and Amnon Meisels. Hybrid search for minimal perturbation in dynamic csps. *Constraints*, 16(3):228–249, 2011.

Chapter 8

A Strategic View on University Timetabling

Michael Lindahl · Andrew Mason · Matias Sørensen · Thomas R. Stidsen

Abstract University Timetabling has traditionally been studied as an operational problem where the goal is to assign lectures to rooms and timeslots and create timetables of high quality for students and teachers. Two other important decision problems arise before this can be solved: what rooms are necessary, and in which teaching periods? These decisions may have a large impact on the resulting timetables and are rarely changed or even discussed. This paper focuses on solving these two strategic problems and investigates the impact of these decisions on the quality of the resulting timetables.

The relationship and differences between operational, tactical and strategic timetabling problems are reviewed. Based on the formulation of curriculum-based course timetabling and data from the Second International Timetabling Competition (ITC 2007), three new bi-objective mixed-integer models are formulated. We propose an algorithm based on the ϵ-method to solve them. The algorithm can be used to analyze the impact of having different resources available on most timetabling problems. Finally, we report results on how the three objectives - rooms, teaching periods and quality - influence one another.

Keywords Timetabling · Multiple objective programming · Integer programming

8.1 Introduction

Educational timetabling has gained a lot of attention within operations research and has been studied for a long time; for an overview see Kingston [2013] and Kristiansen and Stidsen [2013]. Many variations of the problem exist as it varies between different educational stages and countries. A definition commonly used for timetabling comes from Wren [1996]:

"Timetabling is the allocation, subject to constraints of given resources to objects being placed in space time, in such a way as to satisfy as nearly as possible a set of desirable objectives."

This definition captures the timetabling problems that are usually investigated in the literature. However, other important related decisions are rarely investigated. For example: Which resources should be available? Where in space and time is it allowed to place these objects? This paper deals with decisions as to the availability of rooms and teaching periods and analyzes how these decisions affects the resulting timetable.

An overview of the decision problems that relate to timetabling are illustrated in Figure 8.1. The value of a timetable obtained by solving these problems is characterized by three measures associated with the timetabling process:

Agility Responding to changes and quickly make alterations to the timetable.

Quality A timetable that gives a good work environment for employees and students.

Cost A timetable which uses costly resources efficiently.

We categorize each of these problems as either operational, tactical or strategic. The strategic problems are the ones that are solved first and which then affect the tactical problems that are solved next. Finally, the operational problems are solved last. The problems we find at each level, as shown in Figure 8.1, are as follows:

Operational The operational problems are the ones that result in final timetable, i.e. deciding which lectures should be taught in what rooms and when. This main problem is solved every semester, and we separate it into two closely related cases. The first case is the *assignments problem*, which is the feasibility problem of finding a conflict-free timetable.

A common addition to this problem, which is the second case, is including quality measures that aim to make a desirable timetable for students and staff, i.e. the *quality problem*. Soft constraints formulate this, and the violation of these should be minimized. Both course timetabling problems for the Second International Timetabling Competition, formulated in Di Gaspero et al. [2007] fall into this second category.

After a timetable is put into production, disruptions will often occur during the semester as alterations need to be made either due to changes. Handling such disruptions requires solving the minimal perturbation problems addressed in Muller et al. [2005], Phillips et al. [2014]. When decision-support tools are used to solve these problems, it provides agility by being able to create the timetable more quickly and responding faster to changes.

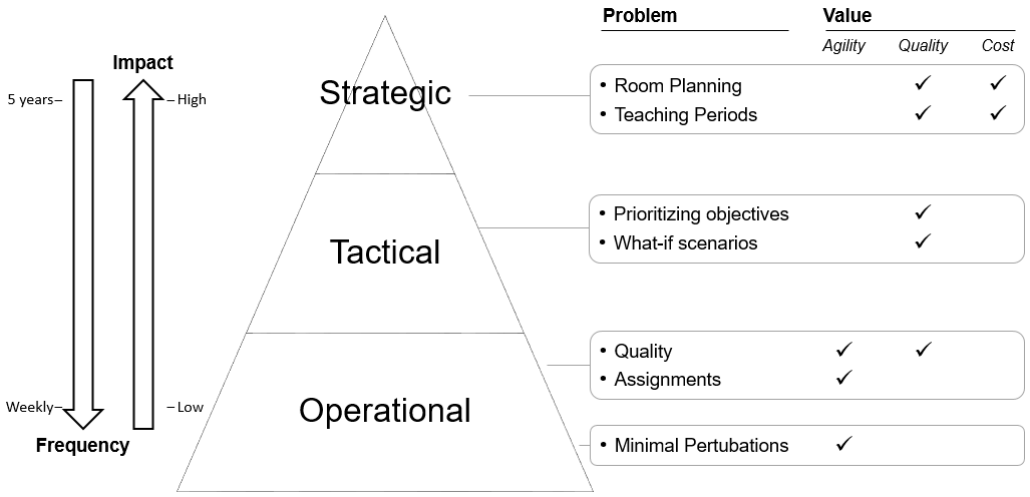


Figure 8.1: Different variations of the timetabling problem occur at different strategic levels. Each problem contributes with different value to the organization when solved. The top problems are long-term decisions that have an impact on all timetables whereas the problems at the bottom have lower impact but are decisions that need to be often taken.

Tactical The tactical problems involve the decisions that are taken before creating the actual assignments. This includes, for example, to prioritize the different quality measures or how to divide the timetabling workload between multiple planners. These decisions affect the quality because a course could end up being taught in an undesired timeslot because the planner does not have other rooms available, even though there may be free rooms in another part of campus.

Strategic The strategic problems are the long-term decisions that affect the timetable and have a high impact on the organization. This could, for example, be to allow teaching to be scheduled later in the day by adding additional timeslots, which we define as the *Teaching Periods* problem. Adding extra timeslots has a high impact on the timetable for professors and students and would therefore often be a decision made by the top management. Buildings and rooms are a high-cost areas for universities because new buildings are expensive and unused rooms can often be turned into offices or rented out. However, these decisions can not be changed from year to year, and it is, therefore, crucial that new rooms are of the actually needed size, and that a room is not rented out for five years if it is required in the next semester. We refer to this problem as the *Room Planning* problem.

Compared to the operational and tactical problems, strategic problems have received little attention in the literature. The *Room Planning* problem was first explored in Fizzano and Swanson [2000], where they show how their assignment model can be used to find the

minimum number of rooms necessary to create a feasible schedule. This is done iteratively by removing rooms until the problem becomes infeasible. In Beyrouthy et al. [2007] they analyze how available rooms affect the utilization and visualize this in multiple ways. They then show how the room profile can be adjusted to achieve a higher utilization. They investigate room robustness by sampling the sizes of the courses under different scenarios. They do not, however, include timetable constraints such as curricula or teachers that can make it impossible to create a timetable that does not conflict with the given rooms.

The authors in Beyrouthy et al. [2009] include timetabling constraints and examine the utilization of rooms by determining how many courses can be planned with a given room profile. They conclude that different constraints contribute to lower the utilization. In Beyrouthy et al. [2010] the authors investigate space type planning, where space types refer to various kinds of rooms such as lecture halls and tutorial rooms. The total capacity is fixed.

The *Teaching Periods* problem of choosing the number of timeslots has not previously been investigated. Decision-support tools for management that support these important decisions are limited, or rather non-existent.

The purpose of this paper is to create new insight into how these strategic decisions affect the timetable and to lay the foundation for future research within this area. We will investigate the two strategic decisions of deciding which rooms to use and how many timeslots there should be. We will also analyze how these decisions affect the quality of the timetable by using bi-objective optimization and mixed-integer programming.

In section 8.2 we define the strategic problems and in section 8.3 we show the method used to solve these problems. The results are shown in section 8.4, and finally conclusions and suggestions for future research are given in section 8.5.

8.2 Curriculum-based Course Timetabling

University Timetabling differs a lot between universities due to differences between education and traditions between countries. Therefore, it has been difficult for researchers to compare their work, as they were solving different problems. To overcome this problem, timetabling competitions have been held where common problem formulations and benchmark instances have been formulated.

As a basis for this analysis, we use the data-sets and problem formulation of Curriculum-based Course Timetabling (CB-CTT) from the Second International Timetabling Competition ITC-2007 stated in Di Gaspero et al. [2007]. For an overview of the research done on this problem, we recommend Bettinelli et al. [2015]. As discussed, this is an operational problem, and we will now give the formulation of the original problem and then show how it can be used as a basis to solve the strategic problems.

The original curriculum-based course timetabling problem is formulated as follows: A set of courses is given, and each course consists of a number of lectures that should be planned. A timeslot and a room should be assigned to each lecture without causing conflict, where a timeslot is a time on a particular weekday. Two lectures from one course cannot take place in the same timeslot, as this will cause a conflict. Besides courses and timeslots, a list of rooms is also given, and only one lecture can be taught in a room in one

timeslot. Each room also has a certain size, and it is a requirement that a room should be able to accommodate all the students following a given course allocated in that room. Each course is associated with a teacher, and a teacher can only teach one course at the time. Finally, we also have a set of curricula that consists of a set of courses that cannot be placed in the same timeslot.

The original formulation includes four different quality measures (soft-constraints): `RoomCapacity`, stating that the room should be able to accommodate all students; `RoomStability` that states that all lectures from the same course should be planned in the same room; `MinimumWorkingDays` ensures that a lecture is spread over a minimum number of days specified in the data, where a penalty is paid for each day not used, and finally, `CurriculumCompactness`, where a penalty is paid if a lecture from a curriculum is not scheduled next to a course from the same curriculum. To create our models, we define the following sets:

$\mathcal{C}$**:** Set of courses

$\mathcal{CU}$**:** Set of curricula

$\mathcal{P}$**:** Set of timeslots across the week

$\mathcal{D} = \{P_{\text{Mo}}, P_{\text{Tu}}, P_{\text{We}}, P_{\text{Th}}, P_{\text{Fr}}\}$, set of timeslots belonging to each day of the week e.g. P_{Mo} is all the timeslots on Mondays.

$\mathcal{R}$**:** Set of rooms

We then have the following parameters:

$l(c)$**:** The number of lectures for course $c \in \mathcal{C}$

$mnd(c)$**:** The minimum number of weekdays on which there must be a lecture for course $c \in \mathcal{C}$

$dem(c)$**:** The demand for course $c \in \mathcal{C}$, i.e. the number of students in that course.

$cap(r)$**:** The capacity of room $r \in \mathcal{R}$

We also define helper sets that are used to formulate the model:

$\mathcal{S} = \{cap(r) : r \in \mathcal{R}\} \cup \{0\}$, set of unique room capacities, including zero

$\mathcal{S}_{\geq s} = \{s' \in \mathcal{S} : s' \geq s\}$, set of room capacities larger than or equal to $s \in \mathcal{S}$

$\mathcal{C}_{\geq s} = \{c \in \mathcal{C} : dem(c) \geq s\}$, set of courses with a demand larger than or equal to $s \in \mathcal{S}$

$\mathcal{R}_{\geq s} = \{r \in \mathcal{R} : cap(r) \geq s\}$, set of rooms with a capacity larger than or equal to $s \in \mathcal{S}$

8.2.1 Quality Problem

The quality problem is the standard problem in timetabling where the goal is to create a feasible timetable of high quality. A number of soft constraints are defined, and the quality is measured by the number of violations of these. Compared to the original description of CB-CCT we make two changes. Because we want to explore the trade-off between the number of rooms, we turn `RoomCapacity` into a hard constraint, meaning that rooms cannot be overbooked with more students than there can be seated. The second change is that the soft objective `RoomStability` is removed. This is to reduce the computational complexity, because, as shown in Lach and Lübbecke [2012], preventing lectures from being assigned to specific rooms can reduce the number of decision variables significantly. The penalty for `RoomStability` is also the smallest of the soft constraints and contributes the least to the objective. Note that this simplification does not alter the general applicability of our method.

$$\min \quad f_{qual} = \sum_{c \in \mathcal{C}} 5 \cdot w_c + \sum_{cu \in \mathcal{CU}, p \in \mathcal{P}} 2 \cdot v_{cu,p} \tag{9a}$$

$$\text{s.t.} \quad \sum_{c \in \mathcal{C}_{\geq s}} x_{c,p} \leq |\mathcal{R}_{\geq s}| \qquad \forall s \in \mathcal{S}, p \in \mathcal{P} \tag{9b}$$

$$\sum_{p \in \mathcal{P}} x_{c,p} = L(c) \qquad \forall c \in \mathcal{C} \tag{9c}$$

$$\sum_{p \in d} x_{c,p} - z_{c,d} \geq 0 \qquad \forall c \in \mathcal{C}, d \in \mathcal{D} \tag{9d}$$

$$\sum_{d \in \mathcal{D}} z_{c,d} + w_c \geq mnd(c) \qquad \forall c \in \mathcal{C} \tag{9e}$$

$$\sum_{c \in \mathcal{CU}} x_{c,p} - q_{cu,p} = 0 \qquad \forall cu \in \mathcal{CU}, p \in \mathcal{P} \tag{9f}$$

$$-q_{cu,p-1} + q_{cu,p} - q_{cu,p+1} - v_{cu,p} \leq 0 \qquad \forall cu \in \mathcal{CU}, p \in \mathcal{P} \tag{9g}$$

$$\sum_{c \in \mathcal{C}(t)} x_{c,p} \leq 1 \qquad \forall t \in \mathcal{T}, p \in \mathcal{P} \tag{9h}$$

$$x_{c,p} \in \mathbb{B} \qquad \forall c \in \mathcal{C}, p \in \mathcal{P} \tag{9i}$$

$$w_c \in \mathbb{R}^+ \qquad \forall c \in \mathcal{C} \tag{9j}$$

$$z_{c,d} \in [0,1] \qquad \forall c \in \mathcal{C}, d \in \mathcal{P}_d \tag{9k}$$

$$q_{cu,p} \in [0,1] \qquad \forall cu \in \mathcal{CU}, p \in \mathcal{P} \tag{9l}$$

$$v_{cu,p} \in [0,1] \qquad \forall cu \in \mathcal{CU}, p \in \mathcal{P} \tag{9m}$$

Model 9: The MIP model for the Quality Problem.

The mixed integer model we use is based on the one proposed in Lach and Lübbecke

[2012] and is seen in Model 9. It includes the following variables and constraints: The decision variable is the binary $x_{c,p}$ that indicates if course $c \in \mathcal{C}$ is planned in timeslot $p \in \mathcal{P}$. Constraint (9c) ensures that all lectures are planned. The constraint (9b) ensures that the rooms are able to accommodate all students in each course.

The two objective terms in (9a) are calculated the following way: To calculate `MinimumWorkingDays` we introduce a positive variable $z_{c,d}$ which takes the value 1 if course $c \in \mathcal{C}$ is planned on day $d \in \mathcal{D}$ and otherwise zero. This is ensured by constraint (9d) and the objective pressure that pushes $z_{c,d}$ to one, if possible. The violation is then calculated by constraint (9e) and the variable w_c. To calculate `CurriculumCompactness` we introduce the positive variable $q_{cu,p}$, which takes the value 1 if curriculum $cu \in \mathcal{CU}$ is planned in timeslot $p \in \mathcal{P}$ and otherwise zero. The positive variable $v_{cu,p}$ then takes the value 1 if there is a violation in curriculum $cu \in \mathcal{CU}$ in timeslot $p \in \mathcal{P}$.

8.2.2 Room Planning Problem

The second problem we will consider is the *Room Planning Problem* where the objective is to find the minimum number of rooms to use to accommodate all courses to be taught. In this section we will introduce the simplest form of this problem, as given in Model 10.

$$\min \quad f_{seats} = \sum_{s \in \mathcal{S}} s \cdot r_s \tag{10a}$$

$$\text{s.t.} \quad |\mathcal{P}| \sum_{s \in \mathcal{S}_{\geq s}} r_s \geq \sum_{c \in \mathcal{C}_{\geq s}} l(c) \quad \forall s \in \mathcal{S} \tag{10b}$$

$$r_s \in \mathbb{Z}^+ \quad \forall s \in \mathcal{S} \tag{10c}$$

Model 10: Set covering MIP-Model for the Room Planning Problem.

Instead of having a fixed set of rooms we introduce a decision variable $r_s \in \mathbb{Z}^+$ that determines the number of rooms of size s that should be used. The capacity of a room is correlated with the physical size, and therefore, also with the cost of a room. Thus, we will seek to minimize the total number of seats as defined in Equation (10a).

It should be noticed that minimizing this metric is the same as maximizing the utilization, a commonly used metric, defined in Beyrouthy et al. [2007] as:

$$\mathit{Utilization} = \frac{\sum_{c \in C} l(c) \cdot dem(c)}{|\mathcal{P}| \sum_{s \in \mathcal{S}} s \cdot r_s}$$

We need to decide which room sizes the model should be able to choose from, i.e. sizes included in the set S. In an optimal room profile, a room will always have the same size as the largest course assigned to it, otherwise it would be wasted capacity. Therefore S should include the sizes of all courses. Because the number of variables, and thus the complexity, is depending on the size of S, it is desirable to keep this set small. In a practical setting it is unlikely that a manager would distinguish between a room of 22 or

25 seats. Consequently, to make the set smaller, we define a parameter δ and only use room sizes that are multiples of the value of this parameter. The largest room will have the size of the largest course rounded up to the nearest multiple of δ. The set S can subsequently be generated in the two following ways:

$$S_{optimal} = \{dem(c) : c \in \mathcal{C}\}$$
$$S_{approx.} = \left\{\delta \cdot \left\lceil \frac{dem(c)}{\delta} \right\rceil : c \in \mathcal{C}\right\}$$

When we decide to use a room, we can plan as many lectures in it as we have timeslots ($|\mathcal{P}|$). To have enough rooms to plan all lectures we define the set covering constraint (10b). The resulting model of the Room Planning Problem provides the same solutions as the one obtained in the single scenario case in Beyrouthy et al. [2007]. They prove that this model can easily be solved to optimality in linear time by using the following greedy algorithm:

1. Sort lectures according to their size, $dem(c)$.
2. While lectures are unassigned to rooms:

   Take the largest unassigned lecture and assign it to a room with free timeslots. If no such room exists, add the smallest room that fits the lecture.

8.2.3 Teaching Periods Problem

Another important factor that impacts on the capacity of a university is the number of timeslots that are available for teaching; this is explored in the Teaching Period problem, shown in Model 11, which decides how many time slots there should be.

$$\min \quad f_{time} = \sum_{p \in \mathcal{P}} t_p \tag{11a}$$
$$\text{s.t.} \quad |\mathcal{R}_{\geq s}| \sum_{p \in \mathcal{P}} t_p \geq \sum_{c \in \mathcal{C}_{\geq s}} l(c) \quad \forall s \in \mathcal{S} \tag{11b}$$
$$t_{p+1} - t_p \leq 0 \quad \forall p \in \mathcal{P} \tag{11c}$$
$$t_p \in \mathbb{B} \quad \forall p \in \mathcal{P} \tag{11d}$$

Model 11: MIP-model Teaching Periods problem.

The mixed-integer model is seen in Model 11. A binary variable, t_p determines if timeslot $p \in \mathcal{P}$ is allowed. We then define the new objective f_{time} to be the total number of used timeslots as defined in (11a).

Similar to the previous model we have a set-covering constraint (11b) which ensures there is enough timeslots, that there is a timeslot and a room of matching size for each lecture.

In practice, timeslots, are chosen so that they are consecutive, as empty timeslots in the middle of the day are unwanted (except for other events like lunch breaks, meetings etc.). Constraint (11c) ensures that a new timeslot can only be used if the previous one is also used. Timeslots are ordered starting from Monday morning with the addition of a new timeslot each day, and then back to Monday; this order is illustrated in Table 8.1.

Mon	Tue	Wed	Thu	Fri
1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
		...		

Table 8.1: The indexing order of the timeslots.

8.2.4 Room Planning vs. Quality

The previous Room Planning model does not consider if it is possible to create a feasible timetable as no timeslots are assigned. Neither does it take the quality of the resulting timetable into account. We will do this in the Room Planning vs. Quality model. Because the available rooms restrict the quality model, it is necessary to take this into account when deciding what rooms to use, otherwise, it might result in a timetable of unacceptable poor quality. To analyze this, we use bi-objective optimization. Bi-objective optimization is a way to explore the trade-off between two objectives by generating multiple so-called pareto-optimal solutions. A pareto-optimal solution is a solution where one of the objectives cannot be improved without making the other worse; for further reading see Ehrgott [2000].

Model 12 is made by merging Model 9 and Model 10. The two models are connected by constraint (12c) which ensures that there can not be more lectures planned in a timeslot than rooms available. The r^+ variables have been added, because experiments show that they help the solver make stronger branches during the branch-and-bound search.

8.2.5 Teaching Periods vs. Quality

Similar to the previous problem, we want to explore the trade off between teaching periods and quality, as it is expected that more timeslots will give more freedom in the planning process. We do this by merging Model 9 and Model 11, which results in Model 13. The two models are connected by constraint (13c) which states that lectures only can be assigned to a timeslot available for use.

$$
\begin{aligned}
\min \quad & f_{seats} : (10a) && && (12a)\\
& f_{qual} : (9a) && && (12b)\\
\text{s.t.} \quad & \sum_{c \in \mathcal{C}_{\geq s}} x_{c,p} - r_s^+ \leq 0 && \forall s \in \mathcal{S}, p \in \mathcal{P} && (12c)\\
& r_s^+ - \sum_{s' \in \mathcal{S}_{\geq s}} r_{s'} = 0 && \forall s \in \mathcal{S} && (12d)\\
& (9c) - (9m) && && (12e)\\
& (10b) - (10c) && && (12f)
\end{aligned}
$$

Model 12: MIP-model for the Room Planning vs. Quality Problem.

8.2.6 Room Planning vs. Teaching Periods

When universities want to increase the number of courses above their current capacity, they can either build more rooms or increase the number of timeslots. In the Room Planning vs. Teaching Periods problem we investigate how these two objectives influence each other. This is done by merging Model 9, Model 12 and Model 13 into the new Model 14.

8.3 Solution Methods

In this section, we propose an algorithm for solving our three bi-objective models. The original quality problem in Model 9 is still hard to solve, and at this time 4 out of the 21 datasets are still not solved to proved optimality; for the current status of this see `http://satt.diegm.uniud.it/ctt/`. The hardness also shows in Cacchiani et al. [2013] where six hours of running time are used for each instance to find good lower bounds. Because of this, we do not expect to solve these problems to optimality when making them bi-objective.

$$
\begin{aligned}
\min \quad & f_{time} : (11a) && && (13a)\\
& f_{qual} : (9a) && && (13b)\\
\text{s.t.} \quad & x_{c,p} - t_p \leq 0 && \forall p \in \mathcal{P}, c \in \mathcal{C} && (13c)\\
& (9b) - (9m) && && (13d)\\
& (11c) - (11d) && && (13e)
\end{aligned}
$$

Model 13: MIP-model for the Teaching Periods vs. Quality Problem.

$$
\begin{aligned}
\min \quad & f_{seats} : (10a) && \text{(14a)}\\
& f_{time} : (11a) && \text{(14b)}\\
\text{s. t.} \quad & (9c) - (9m) && \text{(14c)}\\
& (11c) - (11d) && \text{(14d)}\\
& (12c) - (12d) && \text{(14e)}\\
& (13c) && \text{(14f)}
\end{aligned}
$$

Model 14: MIP-model for the Rooms vs. Teaching Periods problem.

Many methods exist for solving bi-objective methods. A popular one is the ϵ-method from Haimes et al. [1971] that has been applied to many different problems.

Both f_{seats} and f_{time} can only take discrete integer values and f_{seats} can only take values that depend on the size of potential rooms. From experiments, we have seen that the span between the minimum and maximum value is relatively small and only a limited set of values of f_{seats} and f_{time} exists. We, therefore, choose to use the ϵ method, where an artificial constraint is put on one of the objectives while minimizing the other. Because of the small set of potential values we only have to solve a small number of MIP's to explore all potential values of interest.

Let f_x be the objective with a limited discrete set of values, in our case f_{seats} or f_{time}, and let f_y be the other objective. The algorithm for solving objective f_x vs. f_y is shown on Algorithm 8.3.1 and illustrated in Figure 8.2. The algorithm first calculates one lexicographic solution and use this to put a feasibility limit on each objective. This done by first finding a lower limit on f_y and then uses it to calculate an upper feasibility limit for f_x. Afterward, the algorithm starts with setting the ϵ-constraint to the minimum value of f_x and then repeatedly steps it up by the amount Δ. These two limits are used to terminate the algorithm when one of them are exceeded. The reason for also having a limit on f_x and not only f_y is that we enforce a timelimit and do not expect to solve each problem to optimality. As shown in Lodi [2013], modern MIP solvers are heuristic and use randomness. Because of this, the algorithm can turn out in a lucky way and find a close to optimal value for min_y, which it would then not be able to find later due to this randomness.

This method is generic for hard bi-objective problems where there is a limited set of values for one objective. Therefore, it can be used with different timetable constraints and quality measures. It should also be noted that if each subproblem is solved to optimality, this method will produce the optimal pareto-front.

8.4 Results

To show the results obtained by the proposed models and solution methods we use the instances from ITC2007 based on real-world examples from the University of Udine. All

Algorithm 8.3.1 ϵ-method - f_x vs. f_y

Parameters: Δ ▷ Discretization of f_x
$min_y \leftarrow Minimize(f_y)$ ▷ Lower limit on f_y
Add constraint: $f_y = min_y$
$max_x \leftarrow Minimize(f_x)$ ▷ Upper limit on f_x
Remove constraint: $f_y = min_y$
$\epsilon \leftarrow Minimize(f_x)$
Add ϵ-constraint: $f_x \leq \epsilon$
repeat
    $(\hat{f}_x, \hat{f}_y) \leftarrow Minimize(f_y)$ ▷ Find pareto-solution
    $\epsilon \leftarrow \epsilon + \Delta$
    Update ϵ-constraint: $f_x \leq \epsilon$
until $\Delta < max_x \wedge \hat{f}_y > min_y$ ▷ Only continue inside limits

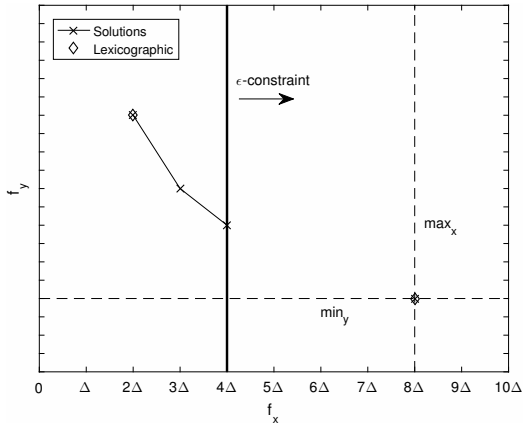


Figure 8.2: An illustration of Algorithm 8.3.1, where the ϵ-constraint moves in steps of Δ. The algorithm terminates when it finds a solution where $\hat{f}_x > max_x$ or $\hat{f}_y < min_y$.

data sets are available from `tabu.diegm.uniud.it/ctt/`. The complete source-code to make the computations are available on `github.com/miclindahl/UniTimetabling`. All computations are made on a 64 bit Windows machine with a 4 GHz Intel Core i7 CPU and 32 GB of memory. To solve the integer programs, we use Gurobi 6.5 with standard settings.

Because this is a strategic problem not often solved and not require in any, urgency the timelimit for each iteration is set to 15 minutes. To generate the set $\mathcal{S}$ in the Room Planning problem, we use $\delta = 25$.

As stated, each course is associated with a number of timeslots in which they cannot be taught. In the Teaching Periods Problem we want to explore what happens when additional timeslots are added. To mark some of these timeslots as unavailable for courses we simulate it by using the following model: The probability that a course is unavailable in a new timeslot is equal to the fraction of timeslots that the course already is unavailable

in. This way we ensure that the new timeslots are realistic and have properties similar to the real data.

In this section, we will first discuss some attributes of the problem that makes it computational difficult and afterward we show results for the three different problems. The trade-offs between the objectives will be reported with plots of the pareto-solutions including bounds and a table with the lexicographic solutions.

8.4.1 Computational Difficulties

Solving the Mixed-Integer programs, shows that the problems are difficult to solve. Two contributors for this is LP solutions and the impact of the ϵ-constraint.

LP-Bound MIP solvers start by relaxing the integer and binary constraints and solving the resulting LP-problem that it uses to find integer solutions. In Figure 8.3 is shown a plot of the solutions for the continuous LP problem and the integer solutions for Model 12. It is seen that there is a large gap between these two solutions. A weak LP relaxation makes it more difficult to find good integer solutions, as the relaxed solutions are used to guide the solver.

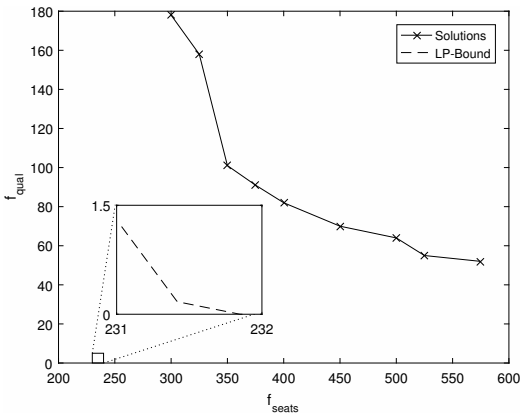


Figure 8.3: Example from `comp18` where it can be seen that the gap between the solution-frontier and the LP-bound is large and does therefore not give much help to the solver.

In Figure 8.3 we observe that for all integer-feasible solutions the LP solution is zero, which implies that the ϵ-constraint does not influence the LP-bound. This is seen in Figure 8.4, where Gurobi's current lower bound on f_{qual} is plotted over time for different ϵ-constraints on f_{seats}. For every value of the ϵ-constraint, the lower bound starts at zero, and it is not until after thirty seconds that the bound differs between the different values of the constraint. This behavior helps explain the long run times we observed.

Threshold Behaviour Another issue which creates difficulties is the so called "threshold behaviour". An example of this is seen in Table 8.2, where `comp18` is solved with three different ϵ-constraints on f_{room}. The original problem with 405 seats available is easy and

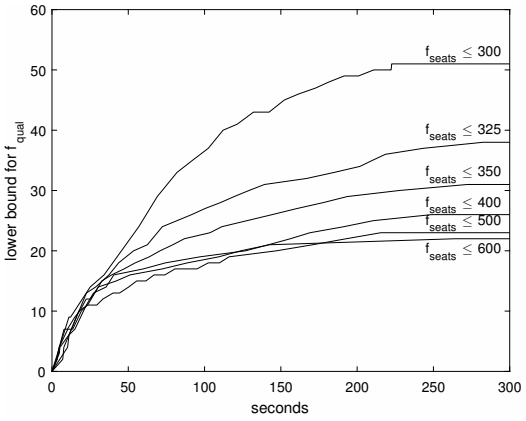


Figure 8.4: Example from `comp18` that shows the bounds improvement over time for different ϵ-constraints. It seen that it starts at zero, and it takes 30 seconds before the ϵ-constraint has a effect on the lower bound.

can be solved to optimality in 33 seconds. We define the critical number of seats, $\tilde{f}_{room}$, as the minimum number of seats that are needed for the problem to be feasible; this can easily be found by minimizing f_{room} in Model 12. If the number of seats is reduced to the critical number, it becomes a lot more difficult and after 30 minutes of run time the gap is still 81%. If the number of available seats is then reduced by only one seat, the problem becomes infeasible and this can be proved in less than a second.

Problem	f_{room}	Time	Gap		Difficulty
Original	405	33s	0%		Easy
$\tilde{f}_{room}$	300	30m	81%	$\Longrightarrow$	Hard
Infeasible	299	0s	-		Easy

Table 8.2: Example of how the threshold effect shows on `comp18`. In the base case the optimal solution can be found in 33 seconds. When setting the maximum number of seats to the critical value, the problem gets difficult and the gap is still 81% after 30 minutes. If one seat is subtracted, the problem is proved infeasible instantly.

This easy-hard-easy behaviour in timetabling is described in Beyrouthy et al. [2008] and is also seen in many other problem types. Beyrouthy et al. [2008] encounters this phase shift when the utilization has a critical value, and, as shown in section 8.2.2, this is directly correlated with f_{seats}. The behaviour is illustrated in Table 8.3. If the number of seats is significant bigger than the critical number of seats, this gives an under-constrained problem that is relatively easy to solve. Usually, heuristics can find good solutions to this problem because there is a lot of freedom to move lectures around without the problem becoming infeasible. In the other case, where the number of seats is smaller than the critical number, the problem is infeasible. This is also an easy problem, as a MIP-solver can prove this infeasibility fast in the pre-solve stage. The last case is the critical-constrained,

which can be seen as *almost infeasible* problems. These are the difficult cases, as heuristics have difficulties in finding good solutions, and MIP-solvers will have trouble with large branch-and-bound trees.

f_{room}	Case	Difficulty	Strategy
$> \tilde{f}_{room}$	Under-Constrained	Easy	Heuristics
$\approx \tilde{f}_{room}$	Critical-Constrained	Hard	?
$< \tilde{f}_{room}$	Over-Constrained	Easy	Prove Infeasibility

Table 8.3: The three cases of behaviour around the threshold described in Beyrouthy et al. [2008]. This shows how the difficulty depends on the ϵ-constraint on f_{room}.

8.4.2 Room Planning vs. Quality

To solve the bi-objective f_{seats} vs. f_{qual} optimization problem we use Model 12 and solve it by using Algorithm 8.3.1 where $f_x = f_{seats}$ is the objective with a limited set of values. To explore all combinations of the given rooms we set $\Delta = 25$. The pareto fronts we generated for all 21 ITC datasets are shown in Figure 8.5 together with the lower bounds returned by the solver; and the two lexicographic solutions are shown in Table 8.4. The total running time of generating the pareto front in minutes is also given.

It is seen that `comp01`, `comp04` and `comp11` only have one solution meaning that there is no benefit from adding extra seats. The other instances only have little quality improvement from adding additional seats. But it can also be seen that the datasets `comp02`, `comp03`, `comp15` and `comp18` have a big trade-off between seats and quality.

The lower bounds from the solver are also shown. It can be seen that the gap is large due to the reasons discussed in section 8.4.1.

Which rooms To get more insights into why a certain solution is chosen as the optimal room configuration we analyze the solutions. A metric for the overcapacity can be deduced from Constraint (10b), allowing us to calculate the $overcapacity_s$ for a given size $s \in \mathcal{S}$ as follows:

$$overcapacity_s = \frac{|\mathcal{P}| \sum_{s \in \mathcal{S}_{\geq s}} r_s}{\sum_{c \in \mathcal{C}_{\geq s}} l(c)} - 1$$

This metric gives the average number of empty timeslots with available rooms for each lecture. A negative number will violate Constraint (10b) and, will therefore, be infeasible. A positive number indicates how much freedom there is to move lectures around. An example of the overcapacity for the solutions is seen in Table 8.5. It is seen that the solver finds solutions that add additional room capacity where the overcapacity is smallest. This metric can be used as a rule of thumb for managers to give them an indication about what rooms to add to increase quality.

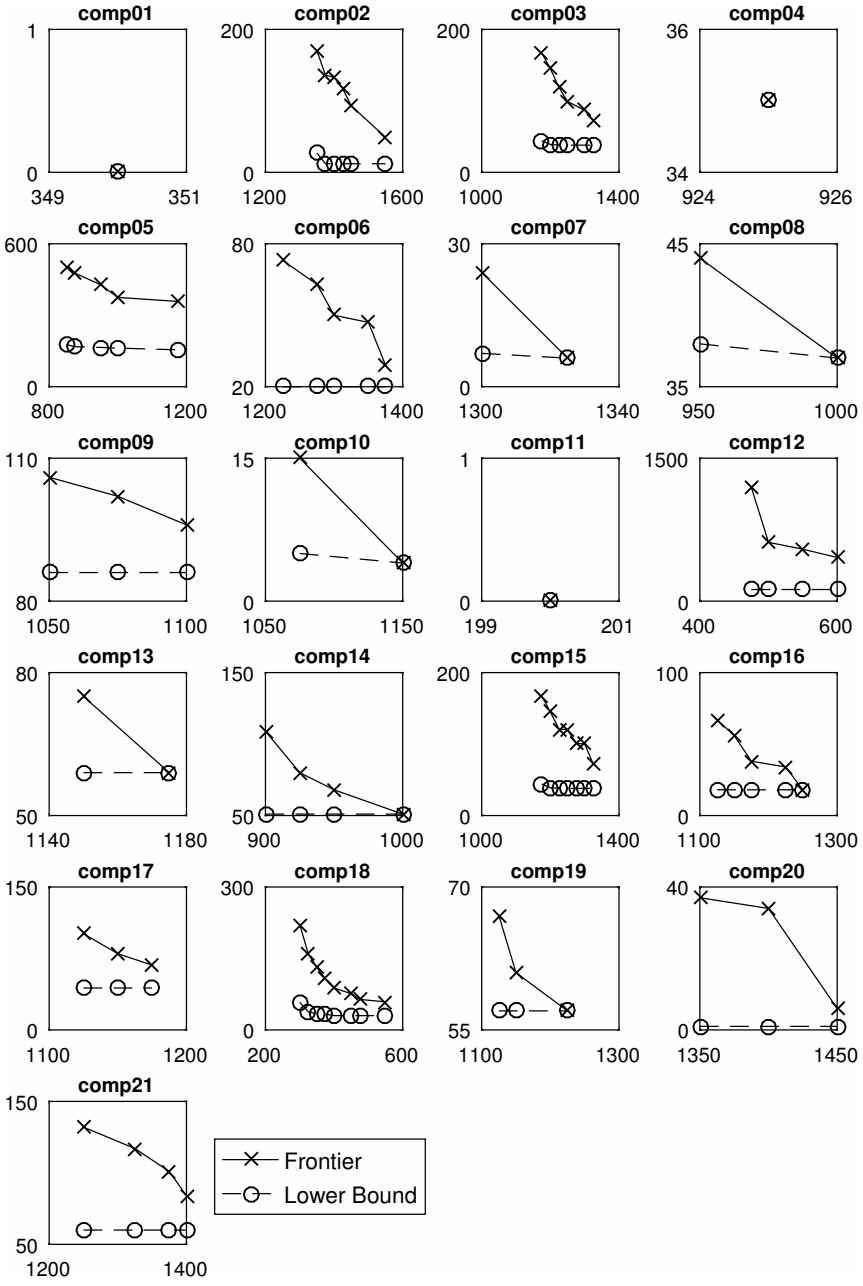


Figure 8.5: The solution frontier for the Room Planning vs. Quality problem. The x-axis is f_{seats} and the y-axis is f_{qual}. In general, the gaps are large. Notice that three of the problems only have one solution.

instance	runtime	fewest seats		best Quality	
	min.	f_{seat}	f_{qual}	f_{seat}	f_{qual}
comp01	0	350	0	350	0
comp02	165	1350	170	1550	49
comp03	120	1175	167	1325	72
comp04	7	925	35	925	35
comp05	226	850	502	1175	358
comp06	120	1225	73	1375	29
comp07	31	1300	24	1325	6
comp08	45	950	44	1000	37
comp09	60	1050	106	1100	96
comp10	75	1075	15	1150	4
comp11	0	200	0	200	0
comp12	105	475	1193	600	458
comp13	33	1150	75	1175	59
comp14	83	900	109	1000	51
comp15	120	1175	167	1325	72
comp16	96	1125	67	1250	18
comp17	60	1125	101	1175	68
comp18	180	300	221	550	59
comp19	91	1125	67	1225	57
comp20	90	1350	37	1450	6
comp21	190	1250	132	1400	84

Table 8.4: The two lexicographic solutions for the Room Planning vs. Quality problem.

	$overcapacity_s$						summary		
f_{seats}	$s =$ 0	25	50	75	100	125	avg.	$\min_1$	$\min_2$
300	0.30	0.33	0.24	1.57	11.00	11.00	4.07	0.24	0.30
325	0.30	0.33	1.48	1.57	11.00	11.00	4.28	0.30	0.33
350	0.83	0.33	0.24	1.57	11.00	11.00	4.16	0.24	0.33
375	0.57	0.78	1.48	1.57	11.00	11.00	4.40	0.57	0.78
400	0.57	1.22	1.48	1.57	11.00	11.00	4.47	0.57	1.22
450	0.83	1.67	1.48	1.57	11.00	11.00	4.59	0.83	1.48
475	0.83	1.67	2.72	1.57	11.00	11.00	4.80	0.83	1.57
500	1.09	2.11	1.48	1.57	11.00	11.00	4.71	1.09	1.48
525	1.09	2.11	2.72	1.57	11.00	11.00	4.92	1.09	1.57
550	1.35	2.11	2.72	1.57	11.00	11.00	4.96	1.35	1.57
575	1.35	2.11	2.72	4.14	11.00	11.00	5.39	1.35	2.11

Table 8.5: The *overcapacity* for different solutions to `Comp18`. It can be seen that average overcapacity (avg.) increases by increasing f_{seats}. $\min_1$ and $\min_2$ denote the smallest and second smallest value. It is seen that, in general, when more seats are added the minimum *overcapacity* is also increased.

8.4.3 Teaching Periods vs. Quality

To analyze the trade-off between f_{time} and f_{qual} we use Model 13 and solve it with Algorithm 8.3.1. We apply the ϵ-constraint on f_{time} and use $\Delta = 1$ so that we add only one timeslot in each iteration to investigate all possibilities. Pareto plots showing the trade-off between f_{time} and f_{qual} for all 21 ITC data sets are shown in Figure 8.6. The two lexicographic solutions are given in Table 8.6 together with the running time. Similar to the previous results, it can be seen that seven of the datasets do not benefit from getting extra timeslots. On the other datasets there are significant quality improvements generated by adding extra timeslots. The lower bounds are closer to the solutions and seven problems are solved to optimality. Some of the instances do, however, still have large gaps.

instance	runtime	fewest timeslots		best quality	
	min.	f_{time}	f_{qual}	f_{time}	f_{qual}
comp01	0	32	8	36	0
comp02	255	22	49	36	30
comp03	45	23	80	23	80
comp04	22	20	48	44	32
comp05	75	33	369	35	344
comp06	315	21	48	39	25
comp07	35	22	6	22	6
comp08	21	21	45	45	32
comp09	58	23	100	24	96
comp10	28	21	8	25	4
comp11	0	40	0	40	0
comp12	180	27	553	36	385
comp13	128	19	70	45	48
comp14	343	20	53	44	43
comp15	45	23	80	23	80
comp16	95	19	48	24	18
comp17	45	23	65	23	65
comp18	195	17	192	27	72
comp19	150	23	58	45	42
comp20	90	24	12	27	8
comp21	45	24	97	24	97

Table 8.6: The lexicographic solutions for the Teaching Periods vs. Quality problem.

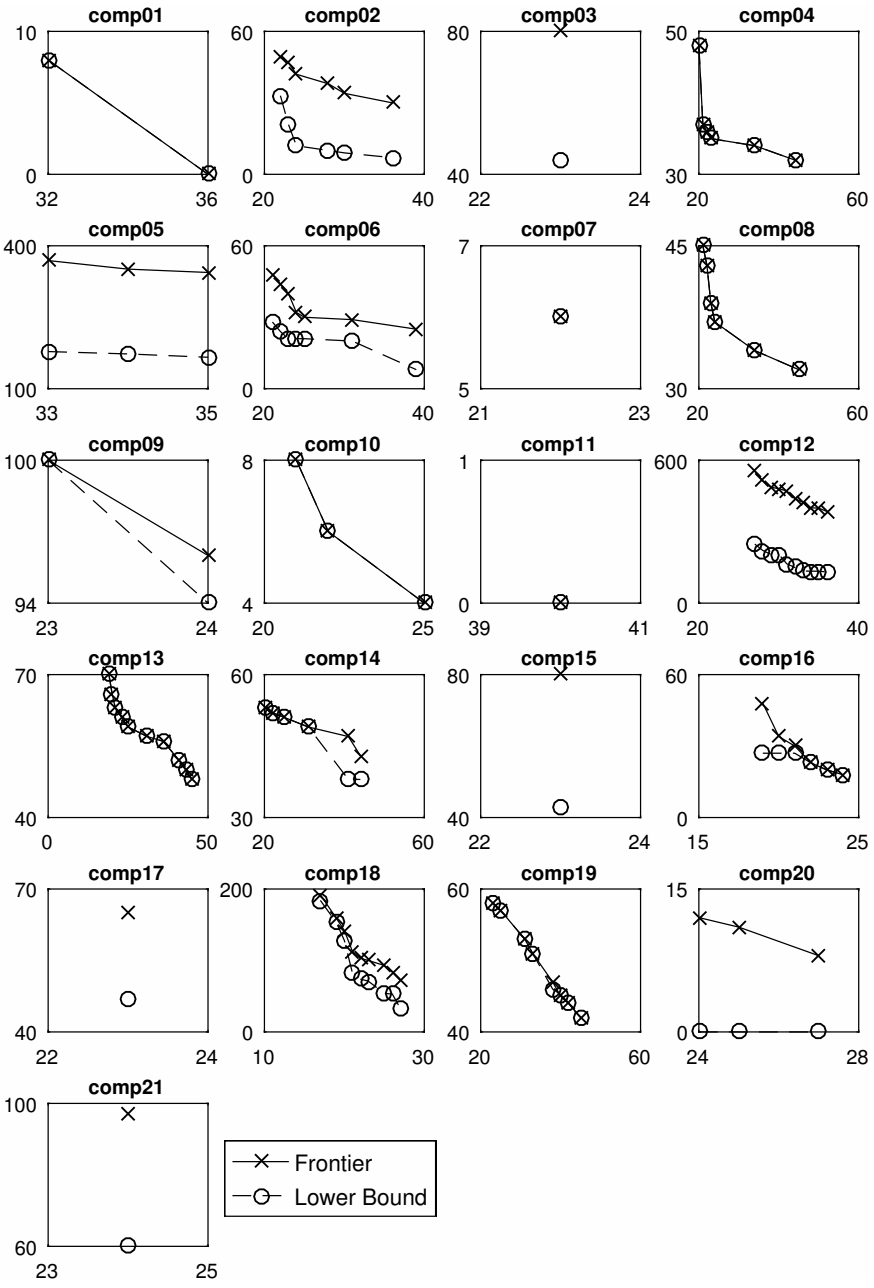


Figure 8.6: The solution frontiers for the Teaching Periods vs. Quality problem. The x-axis is f_{time} and the y-axis is f_{qual}. The gaps are smaller than in the Room Planning vs. Quality problem. Six of the instances only have one pareto-optimal solution.

8.4.4 Teaching Periods vs. Room Planning

To analyze how the two objectives, f_{time} and f_{seats} affect each other we show the trade-off by using the Model 14.

Figure 8.7 shows the pareto front and the two lexicographic solutions, and the running time is shown in Table 8.7. It is seen that this problem is faster to solver, and that optimal pareto frontiers can be generated in less than a minute for all instances. This also shows that the soft constraints has the biggest impact on the difficulty of solving these problems.

The trade-off between f_{time} and f_{seats} is significant on all datasets. Because the utilization U is linear proportional with $|\mathcal{P}| \sum_{r \in \mathcal{R}} cap(r)$ the trade-off is almost linear.

instance	runtime	fewest timeslots		fewest seats	
	min.	f_{time}	f_{seat}	f_{time}	f_{seat}
comp01	0	24	400	54	250
comp02	0	22	1550	49	775
comp03	0	23	1250	50	700
comp04	0	20	1125	48	525
comp05	0	33	850	54	600
comp06	0	18	1625	48	675
comp07	0	20	1575	50	675
comp08	0	21	1125	48	550
comp09	0	23	1100	47	575
comp10	0	19	1375	47	625
comp11	0	28	275	55	150
comp12	0	27	575	59	300
comp13	0	18	1525	50	650
comp14	0	20	1100	46	525
comp15	0	23	1250	50	700
comp16	0	19	1475	46	625
comp17	0	23	1200	49	625
comp18	0	17	475	53	225
comp19	0	23	1175	47	625
comp20	0	18	1800	49	750
comp21	0	24	1250	48	700

Table 8.7: The lexicographic solutions of the results of the Teaching Periods vs. Room Planning Problem.

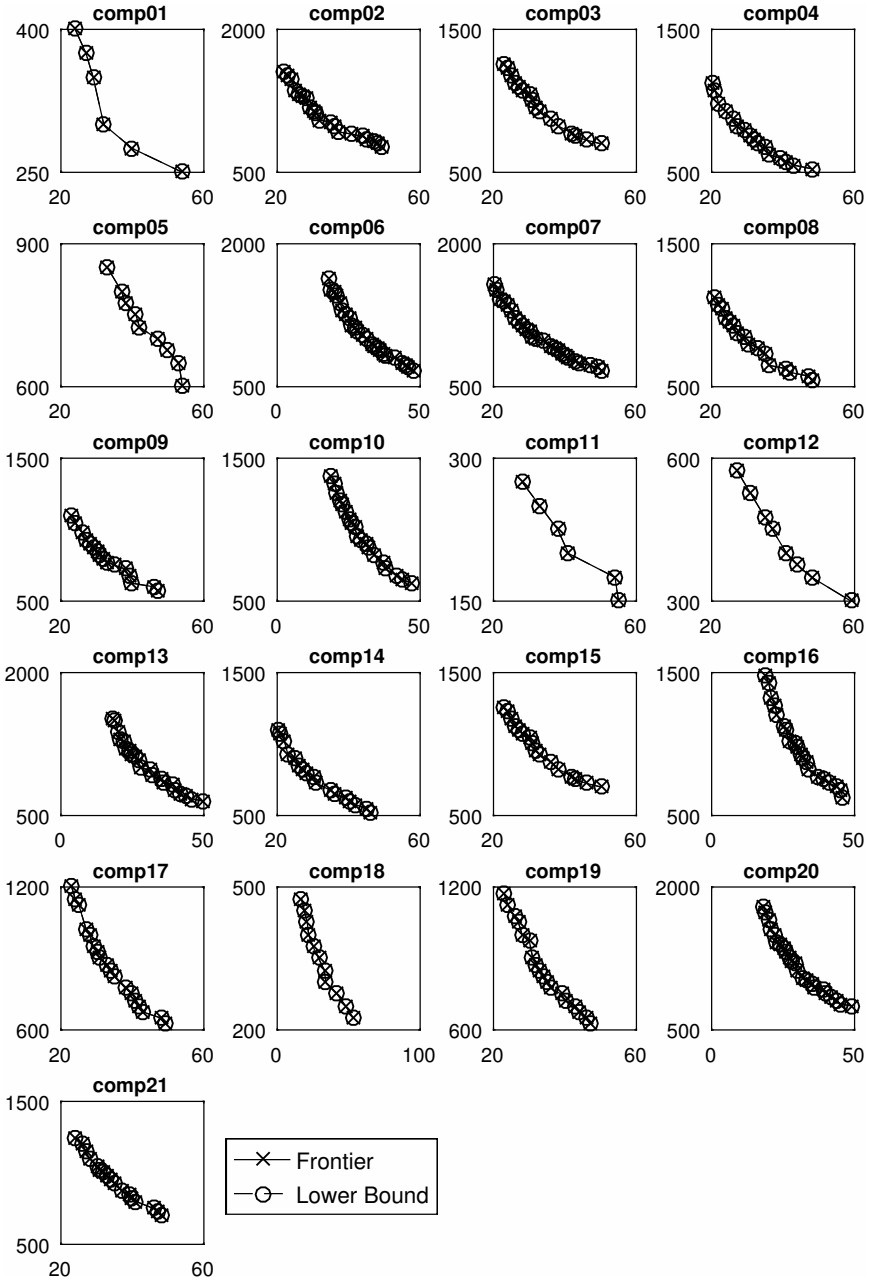


Figure 8.7: The solution frontiers for the Teaching Periods vs. Room Planning Problem. The x-axis is f_{time} and the y-axis is f_{seats}. Notice that all of these solutions are optimal. There is an almost linear relationship between these two objectives.

8.5 Conclusions

University management needs to continuously manage their resources efficiently. Resources like teachers, rooms and teaching periods cannot be increased or decreased on short notice. These management decisions have a high impact on the cost of running a university and also on how good the resulting timetables will be.

Analyzing how strategic decisions affect timetabling quality is both important to help make these high-impact decisions better and to get a better understanding of the structure of the timetabling problem and see how the availability of these resources affects the solution.
This paper presents the first try to investigate these trade-offs. We propose a method that can potentially solve these problems to optimality and create the entire frontier. Our method is applicable to most timetabling problems.

Finally, we have shown how the three objectives - rooms, teaching periods, and quality - have an impact on one another, and we have shown that this interaction between competing objectives varies between the different instances. We have analyzed optimal room profiles to give managers a rule of thumb that can be used to help make decisions better.

Future research This paper explored two soft constraints as quality metrics that both focused on the timeslots in which lectures were planned. Exploring these trade-offs with other metrics could be interesting. Another simplification is that all rooms are considered to be identical except for capacity. In practice, rooms often have certain attributes requested by some courses. Many campuses are so large that the location of the rooms matters; this would give some different trade-offs that would be of interest.

The authors hope that this paper can generate interest in finding and answering more of the strategic questions that are occurring in timetabling.

Acknowledgement

The authors would like to thank the organizers of ITC-2007 for providing a formal problem description of CB-CTT as well as benchmark instances. The authors would also like to thank Alex Bonutti, Luca Di Gaspero and Andrea Schaerf for creating and maintaining the website for instances and solutions to CB-CTT.

Bibliography

Andrea Bettinelli, Valentina Cacchiani, Roberto Roberti, and Paolo Toth. An overview of curriculum-based course timetabling. *TOP*, pages 1–37, 2015.

Camille Beyrouthy, Edmund K Burke, Dario Landa-Silva, Barry McCollum, Paul McMullan, and Andrew J Parkes. Improving the room-size profiles of university teaching space. In *Dagstuhl Seminar on" Cutting, Packing, Layout and Space Allocation," March*. Citeseer, 2007.

Camille Beyrouthy, Edmund K Burke, Dario Landa-Silva, Barry McCollum, Paul McMullan, and Andrew J Parkes. Threshold effects in the teaching space allocation problem with splitting. Technical report, University of Nottingham, 2008.

Camille Beyrouthy, Edmund K Burke, Dario Landa-Silva, Barry McCollum, Paul McMullan, and Andrew J Parkes. Towards improving the utilization of university teaching space. *Journal of the Operational Research Society*, 60(1):130–143, 2009.

Camille Beyrouthy, Edmund K Burke, Barry McCollum, Paul McMullan, and Andrew J Parkes. University space planning and space-type profiles. *Journal of Scheduling*, 13 (4):363–374, 2010.

V. Cacchiani, A. Caprara, R. Roberti, and P. Toth. A new lower bound for curriculum-based course timetabling. *Computers & Operations Research*, 40(10):2466 – 2477, 2013. ISSN 0305-0548.

Luca Di Gaspero, Barry McCollum, and Andrea Schaerf. The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3). Technical report, School of Electronics, Electrical Engineering and Computer Science, Queenes University SARC Building, Belfast, United Kingdom, 2007.

M. Ehrgott. *Multicriteria Optimization*. Springer, 2000.

Perry Fizzano and Steven Swanson. Scheduling classes on a college campus. *Computational optimization and applications*, 16(3):279–294, 2000.

Yacov Y Haimes, LS Ladson, and David A Wismer. Bicriterion formulation of problems of integrated system identification and system optimization, 1971.

Jeffrey H. Kingston. Educational timetabling. In A. Sima Uyar, Ender Ozcan, and Neil Urquhart, editors, *Automated Scheduling and Planning*, volume 505 of *Studies in Computational Intelligence*, pages 91–108. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-39303-7.

Simon Kristiansen and Thomas R. Stidsen. A comprehensive study of educational timetabling - a survey. Technical Report 8, 2013, DTU Management Engineering, Technical University of Denmark, November 2013.

G. Lach and M. Lübbecke. Curriculum based course timetabling: new solutions to udine benchmark instances. *Annals of Operations Research*, 194:255–272, 2012. ISSN 0254-5330.

Andrea Lodi. The heuristic (dark) side of mip solvers. In El-Ghazali Talbi, editor, *Hybrid Metaheuristics*, volume 434 of *Studies in Computational Intelligence*, pages 273–284. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-30670-9.

Tomáš Muller, Hana Rudová, and Roman Barták. Minimal perturbation problem in course timetabling. In Edmund Burke and Michael Trick, editors, *Practice and Theory of Automated Timetabling V*, volume 3616 of *Lecture Notes in Computer Science*, pages 126–146. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-30705-1.

Antony E. Phillips, Cameron G. Walker, Matthias Ehrgott, and David M. Ryan. Integer programming for minimal perturbation problems in university course timetabling. In *10th International Conference of the Practice and Theory of Automated Timetabling PATAT 2014, 26-29 August 2014, York, United Kingdom*, 2014.

A. Wren. Scheduling, timetabling and rostering a special relationship? In Edmund Burke and Peter Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 46–75. Springer Berlin / Heidelberg, 1996.